ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación :

INGENIERO TÉCNICO EN INFORMÁTICA DE GESTIÓN

Título del proyecto:

USOS DE LA LÓGICA DIFUSA E INTERVALOS EN EL RECONOCIMIENTO DEL HABLA

Juan Cerrón González

Raúl Orduna Urrutia

Pamplona, 28-07-2011

INDICE

Resumen

Este trabajo explora el uso de técnicas provenientes de la lógica difusa, para conseguir un sistema de clasificación de fonemas sobre espectrogramas de voz. El motor de clasificación obtenido, dispone de muchos parámetros que se han autoajustado por medio de algoritmos genéticos. El clasificador lingüístico de fonemas obtenido, permite realizar la tarea de reconocimiento del habla con una probabilidad muy alta de acierto.

La principal aportación del trabajo ha consistido en la generalización de un sistema de reconocimiento de habla que utiliza conjuntos difusos con dos formas diferentes de aplicar intervalos, en primer lugar a los datos de entrada, con lo que evitamos perder información en el tratamiento de los datos de entrada, y en segundo lugar en el uso de conjuntos intervalo-valorados difusos en el propio motor de reconocimiento.

1. Introducción

El **Reconocimiento Automático del Habla** (RAH) es un campo de la Inteligencia Artificial pendiente de investigación y desarrollo. A pesar de los múltiples tratamientos del problema que se han realizado, precisión y velocidad no se han podido obtener simultáneamente. Sigue existiendo una diferencia cualitativamente importante entre los resultados obtenidos por medios informáticos y los obtenidos por expertos humanos. En la literatura sobre el reconocimiento automático de voz, se han estudiado diferentes modelos con origen en diferentes campos de la investigación: los *Modelos Ocultos de Márkov* provenientes de la estadística (HMM), las *Redes Neuronales* provenientes de la inteligencia artificial, el *Análisis de Componentes Principales* de la matemática (ACP), etc. Entre estos métodos existen algunos que utilizan, como concepto, un razonamiento similar al del cerebro humano mediante el uso de lógica difusa. Creemos que esta aproximación es más flexible y permite abordar el problema desde diferentes puntos de vista con muy poco esfuerzo.

Las técnicas de lógica difusa utilizadas, han sido probadas con éxito en el análisis de imágenes. Por ejemplo, el grupo de investigación del Departamento de Automática y Computación de la Universidad Pública de Navarra (*Utilización de la lógica difusa en procesamiento de imágenes digitales; Aplicación de los conjuntos intervalo-valorados fuzzy al tratamiento de imágenes de manchas en la piel para diagnosticar melanomas.*), lo utiliza en varios de sus trabajos. Sabemos que se pueden aplicar estas técnicas sobre el análisis de voz, porque el espectrograma (representación de un sonido) puede interpretarse como un tipo de imagen con ciertas restricciones (una de las dimensiones de la imagen corresponde a una magnitud temporal en el mundo físico).

Para la realización del trabajo se ha implementado el artículo de ejemplo de la Universidad de Sharif, *Recognition of human speech phonems using a novel fuzzy approach* (2006). En este artículo se usaba la frase de ejemplo, *She has her dark suit in greasy wash water all year*, obtenida del *linguistic data consortium (LDC).* Para confirmar que los datos que se obtenían en el ejemplo coincidían con los nuestros, se ha usado la misma frase de ejemplo a lo largo de todo el proyecto.

Los resultados han sido comparados con los presentados en los trabajos "*Recognition of human speech phonems using a novel fuzzy approach" (*Halavati, Bagheri Shouraki, Harati Zadeh, 2006) y "*A Novel fuzzy Approach to Speech Recognition"* (Halavati, Bagheri Shouraki, Eshraghi, Alemzadeh, Ziaie, 2004) también centrados en la lógica difusa y cuya principal diferencia es el no uso de intervalos.

Ha resultado difícil entender algunos de los métodos utilizados en el artículo de referencia, debido a que se detectaban incoherencias entre las explicaciones proporcionadas y las imágenes que representaban esas explicaciones. Por esta razón se han realizado algunos cambios con respecto a la implementación exacta del artículo, intentando aproximarnos a los resultados que se observan en los gráficos proporcionados. Estos cambios están detallados en su apartado correspondiente a lo largo del documento.

En los siguientes capítulos del trabajo presentamos una breve explicación de varios de los conceptos más significativos y necesarios para la correcta comprensión del trabajo, una descripción completa de los datos que se van a tratar y del espacio de trabajo a usar, la explicación del proceso de reconocimiento, el entrenamiento al que son sometidos los datos para un mejor reconocimiento final y la especificación de los experimentos realizados, con sus resultados y las conclusiones obtenidas.

2. Antecedentes

2.1. Nociones y conceptos básicos de la Lógica Difusa

El aspecto central de los sistemas basados en la teoría de la **Lógica Difusa** es que, a diferencia de los que se basan en la lógica clásica, tiene la capacidad de reproducir aceptablemente los modos usuales del razonamiento, considerando que la certeza de una proposición es una cuestión de grado. Mas formalmente se puede decir que si la lógica es la ciencia de los principios formales y normativos del razonamiento, la lógica difusa o borrosa se refiere a los principios formales del razonamiento aproximado, considerando el razonamiento preciso (lógica clásica) como caso limite. Así pues, las características más atractivas de la lógica difusa son su flexibilidad, tolerancia con la imprecisión, su capacidad para modelar problemas no lineales y su base en el lenguaje natural.

Un **Conjunto** refleja la tendencia a organizar, resumir y generalizar el conocimiento sobre los objetos del mundo real. El encapsulamiento de los objetos es una colección cuyos miembros comparten una serie de características o propiedades que implican la noción de conjunto. Los conjuntos introducen una noción fundamental de *dicotomía*. En esencia, cualquier proceso de *dicotomía* es una clasificación binaria: o se acepta o se rechaza que un objeto pertenece a una categoría determinada. Normalmente la decisión de *aceptar* se denota por "1" y la de *rechazar* por "0". Esto indica que una decisión de clasificación puede expresarse a través de una función característica.

Sea A un conjunto en el universo X, la función característica asociada a A, A(x), $x \in X$, se define como:

$$A(x) = \begin{cases} 1, & si \quad x \in \mathbf{A} \\ \mathbf{0}, & si \quad x \notin A \end{cases}$$

La Teoría de Conjuntos Difusos es introducida por Zadeh en 1965. Su interés se centra esencialmente en modelar aquellos problemas en donde los enfoques clásicos de Teoría de Conjuntos y Teoría de Probabilidades resultan insuficientes o no operativos. Para ello esta teoría generaliza la noción de conjunto haciéndolo más difuso (*fuzziness*). Los **Conjuntos Difusos** surgen como una nueva forma de representar la imprecisión y

la incertidumbre. La investigación sobre dichos conjuntos ha seguido dos vertientes principales que se complementan: la teoría matemática formal que generaliza y amplía los conceptos de otras áreas de la matemática, y el modelado de sistemas en los que se trata con la incertidumbre; visión automática, teoría de sistemas, teoría de decisión, etc.

Un conjunto difuso pueden contener elementos de forma parcial. Cada elemento tiene un valor entre 0 (exclusión completa) y 1 (pertenencia completa). Este valor expresa el grado con el que el objeto es compatible con las propiedades y características distintivas de una colección. Cuanto más cerca esté del 1, mayor será su pertenencia a esa colección y cuanto más cercano a 0, indicará menor pertenencia.

Un conjunto difuso A sobre X está caracterizado por una función de pertenencia que transforma los elementos de un dominio, espacio, o universo del discurso X en el intervalo [0, 1].

$$\mu_A : X \rightarrow [0, 1]$$

La eficacia de conjuntos difusos depende de lo representativa que sea la función de pertenencia. En muchas ocasiones la elección de este valor es problemática. En estos casos puede ser apropiado representar el valor de pertenencia de un elemento a un conjunto mediante un intervalo de valores en lugar de un sólo valor. A partir de estas ideas surgen los **conjuntos difusos intervalo-valorados** (interval-valued fuzzy sets, IVFSs). IVFSs asignan un subintervalo del intervalo [0,1] a cada elemento del universo. Permiten modelar tanto la vaguedad y carencia de nitidez de los límites de un conjunto, como la imprecisión debida a la incertidumbre y carencia de información. En muchas aplicaciones, IVFSs ofrece resultados menos específicos pero más realistas. La representación formal de un intervalo queda definida por los siguientes axiomas:

Sea $L = \{ [x_1, x_2] \in [0, 1]^2 \text{ con } x_1 \leq x_2 \}$.

1. $[x_1, x_2] \leq_L [y_1, y_2]$ si y solo si $x_1 \leq y_1$ y $x_2 \leq y_2$
2. Un conjunto intervalo-valorado difuso A en un universo X es una función:

$$A = \{(a, [x_1, x_2]) \mid a \in X, [x_1, x_2] \in L\}$$

3. Sea X un universo A y B dos conjuntos intervalo-valorados. La igualdad entre A y B se define:

$$A =_L B \text{ si y solo si } A(a) =_L B(a) \ \forall a \in X.$$

4. Sea X un universo y A y B dos conjuntos intervalo-valorados difusos. La inclusión de A en B se define:

$$A \subseteq_L B \text{ si y solo si } A(a) \leq_L B(a) \ \forall a \in X.$$

Sea $\mu_A(x)$ una función de pertenencia de un conjunto intervalo-valorado se define de la siguiente manera:

$$\mu_A(x) = \begin{cases} cte, & si \ \ \mathbf{x_1} \leq x \leq x_2. \\ \mathbf{0}, & en\ caso\ contrario \end{cases}$$

si μ (a) = [x_1, x_2]

Cualquier forma de la función $\mu_A : X \rightarrow [0, 1]$, describe una **función de pertenencia** asociada a un conjunto intervalo-valorado difuso A. Sin embargo, para que una función de pertenencia represente adecuadamente al conjunto, no es suficiente con representar el concepto del conjunto sino que se debe adecuar al contexto de la aplicación específica que se desea tratar.

Las gráficas de las funciones pueden tener diferentes formas, y las funciones pueden tener propiedades específicas (Ej.: continuidad) que sean críticas en el contexto de la aplicación. Sin embargo, en muchas ocasiones la semántica capturada por los conjuntos difusos no es sensible a variaciones en la forma. En estos casos son convenientes funciones simples para representar la función de pertenencia.

En muchos casos prácticos, los conjuntos difusos pueden representarse con familias de funciones paramétricas. Las más comunes son las siguientes:

- *Función Triangular:*

$$\mu_A(x) = \begin{cases} \mathbf{0}, si \ \ x \leq a \\ \dfrac{x-a}{b-a}, si\ \mathbf{x} \in [\mathbf{a}, \mathbf{b}] \\ \dfrac{c-x}{c-b}, si\ \mathbf{x} \in [\mathbf{b}, \mathbf{c}] \\ \mathbf{0}, si \ \ x \geq c \end{cases}$$

- *Función Trapezoidal (*)*:

$$\mu_A(x) = \begin{cases} 0, si \;\; x \leq a \\ \dfrac{x-a}{b-a}, si \; \mathbf{x} \in [\mathbf{a}, \mathbf{b}] \\ \mathbf{1}, si \; \mathbf{x} \in [\mathbf{b}, \mathbf{d}] \\ \dfrac{c-x}{c-d}, si \; \mathbf{x} \in [\mathbf{d}, \mathbf{c}] \\ \mathbf{0}, si \;\; x \geq c \end{cases}$$

Donde b y d indican el intervalo donde la función de pertenencia vale 1.

(*) Esta función está implementada en el anexo con el nombre de *generalizada()*

- *Función Gaussiana:*

$$A(x) = \mathbf{e}^{-\mathbf{k}(\mathbf{x}-\mathbf{m})^2},$$

Donde k > 0.

En los diferentes problemas que se pueden encontrar en el mundo real, la información que se maneja puede tener diferentes rangos de valoración y los valores pueden tener distinta naturaleza. En ocasiones, la información que manipula un problema puede que no sea fácil de definir de forma precisa mediante un valor cuantitativo (número), sin embargo, puede ser fácilmente valorada en forma cualitativa. En este caso, suele ocurrir que el uso de un enfoque lingüístico difuso se adapte mejor que un enfoque numérico. Esto implica que a la hora de evaluar fenómenos relacionados con la percepción subjetiva (diseño, gusto, …) se suelen utilizar palabras del lenguaje natural (bonito, feo, dulce, salado,…) en lugar de valores numéricos.

En la Teoría de Conjuntos Difusos se pueden representar los aspectos cualitativos como valores lingüísticos mediante *etiquetas lingüísticas***.**

Una **Etiqueta Lingüística** se caracteriza por un *valor sintáctico* (nombre que la identifica) y por un *valor semántico* (que la representa). El nombre que la define es una palabra o frase perteneciente a un conjunto de términos lingüísticos y el significado de dicho nombre viene dado por un subconjunto difuso que lo representa. Al ser las palabras menos precisas que los números, el concepto de etiqueta lingüística parece una buena propuesta para caracterizar a aquellos fenómenos que son demasiado complejos o están mal definidos para poder ser evaluados mediante valores numéricos precisos.

Definido formalmente, es un terceto (*Nombre*, A, X) donde *Nombre* es el nombre asociado al conjunto difuso *A* en el universo *X*. En nuestro documento, las etiquetas están definidas por una escala cromática formada por los colores: negro, azul, rojo, amarillo, verde cian y blanco, que representa valores de intensidad, y una escala de magnitudes que representan las longitudes: muy corto, corto, medio, largo, muy largo. Un ejemplo sería (Negro, conjunto de *valores entre 0 y 0.35*, todos los valores *del intervalo [0, 1]).*

Una **función de agregación** de dimensión n (función de agregación n-aria) es una aplicación creciente $M: [0,1]^n \rightarrow [0,1]$ tal que $M(0, \dots, 0) = 0$ y $M(1, \dots, 1) = 1$.

Sea $M: [0,1]^n \rightarrow [0,1]$ una función de agregación n-aria.

(i) M se dice que tiene un aniquilador (destructor) $a \in [0,1]$, si $M(x_1, \dots, x_n) = a$, para cualquier $a \in \{ x_1, \dots , x_n\}$.

(ii) M se denomina estrictamente creciente si es estrictamente creciente como una función real de n-variables en $[0,1]^n$, si M no tiene aniquilador; y en el dominio $([0,1] \setminus \{a\})^n$ si M tiene un aniquilador a.

(iii) M se dice que tiene divisores de cero si existe un $x_1, \dots , x_n \in [0,1]$ tal que $M(x_1, \dots , x_n) = 0$.

(iv) M se denomina idempotente si $M(x, \dots , x) = x$ para cualquier $x \in [0,1]$.

Para el caso particular de agregaciones de dos variables recordamos las siguientes definiciones.

(i) M se denomina simétrica si $M(x,y) = M(y,x)$ para cualquier $x,y \in [0,1]$.

(ii) M se denomina asociativa si M (M (x,y),z) = M (x, M (y,z)) para cualquier x,y,z ∈ [0,1].

A continuación presentamos algunas de las agregaciones más utilizadas.

Una **norma triangular** (**t-norma**) T: $[0,1]^2 \rightarrow [0,1]$ es una función de agregación bivariable asociativa, conmutativa, y no decreciente tal que T (1,x) = x para todo x [0,1]. Se utiliza mucho en lógica difusa para definir la *intersección* entre conjuntos difusos.

Las tres t-normas básicas más utilizadas son las siguientes: el mínimo o intersección T_M (x,y) = min (x,y), el producto T_P (x,y) = x·y y la t-norma de Lukasiewicz T_L (x,y) = max (x + y – 1,0).

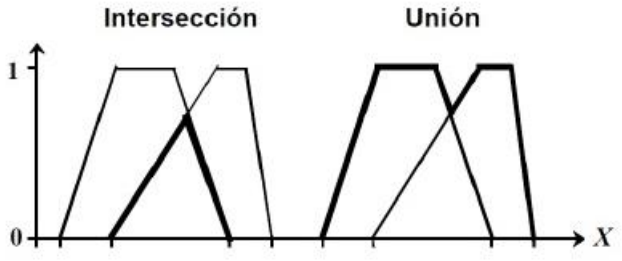


Figura 1. Ejemplo de intersección y unión del mismo conjunto que hacen referencia a la función de las t-normas y las t-conormas.

Cuando trabajemos con varios intervalos, la función mínimo quedará de la siguiente manera: $T_{MI}([x_1, x_2], [y_1, y_2]) = \{\min([x_1, y_1]), \min([x_2, y_2])\}$.

La operación de *unión* de conjuntos para conjuntos difusos puede ser representada por una clase de funciones binarias llamadas **conormas triangulares** o conormas T (**t-conormas**). Toda t-conorma satisface las siguientes desigualdades:

∀ a,b ∈ [0,1]

$u_{max}(a,b) \leq u(a,b) \leq u_{sup}(a,b)$

Las t-conormas básicas más utilizadas son las siguientes: el máximo o unión estándar S_M (x,y) = max (x,y), la suma algebraica S_{Sal} (x,y) = x+y - x·y y la suma acotada o límite S_{ac} (x,y) = min (1, a+b).

Cuando trabajemos con varios intervalos la función máximo quedará de la siguiente manera: $S_{MI}([x_1, x_2], [y_1, y_2]) = \{\max([x_1, y_1]), \max([x_2, y_2])\}$.

2.2. Representación de datos y aproximación a frecuencias humanas

El **espectrograma**, es una representación visual de las diferentes frecuencias que componen un intervalo de tiempo. Para cada instante representado en el eje horizontal, existe una intensidad en cada uno de los valores del eje vertical. A diferencia de la típica señal de representación acústica, en la que sólo se aprecia la amplitud de onda para cada periodo de tiempo, el espectrograma aporta información referente a la frecuencia del sonido.

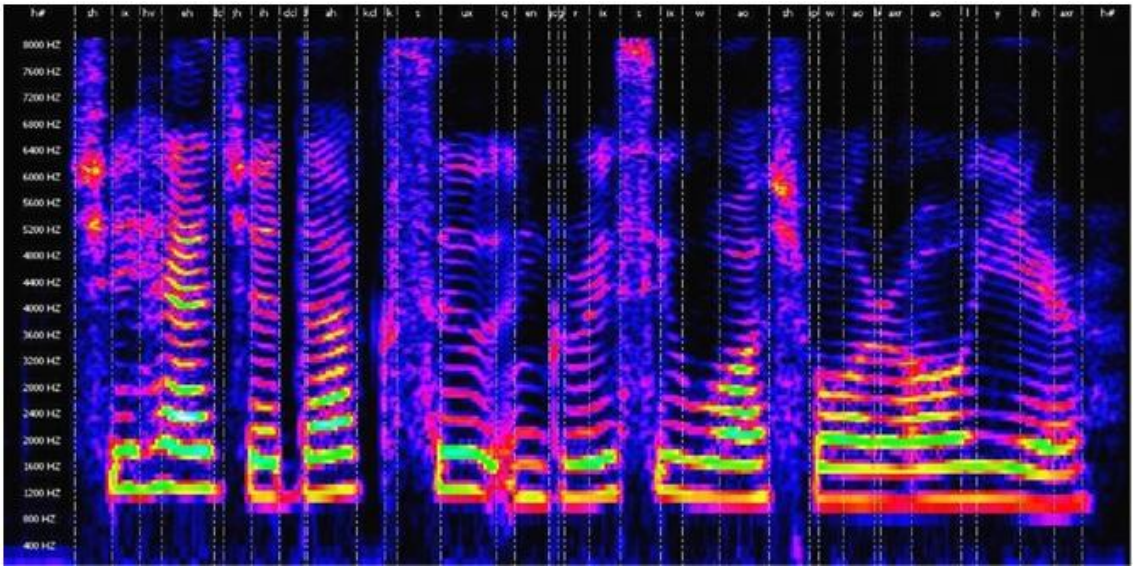


Figura 2. Espectrograma de señales de voz que se obtiene usando 512 muestras para la transformada de Fourier sobre la grabación de ejemplo.

El camino seguido para la construcción del espectrograma ha sido aplicar la *Transformada de Fourier de tiempo Reducido* (STFT) sobre las distintas señales de tiempo observadas en el eje horizontal de la Figura 3.

La función de las *Series de Fourier* es:

$$f(x) = a_0 + \sum_{n=1}^{\infty} \left(a_n \cos \frac{n\pi x}{L} + b_n \sin \frac{n\pi x}{L} \right)$$

(Más información a través del link proporcionado)

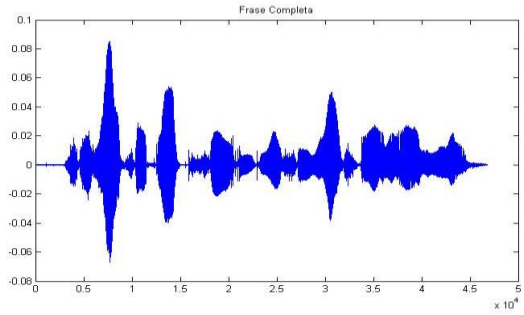


Figura 3. Representación de la grabación de ejemplo en frecuencias y amplitudes de onda.

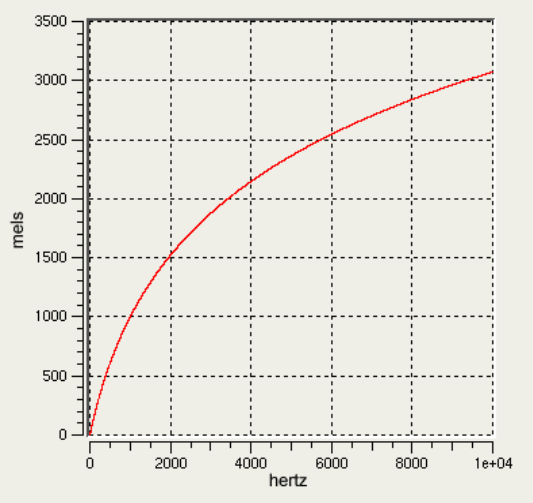


Figura 4. Curva de representación de la escala Mel con las frecuencias en Hzs en el eje horizontal y las frecuencias en Mels en el eje vertical.

La escala de **Mel** fue creada en 1937 por Stevens, Volkman y Newman, cuyo nombre proviene de la palabra *Melody* (melodía). El objetivo de esta escala es situar de forma lineal aquellos tonos que son percibidos por un observador humano como equidistantes. Al ser una escala definida a partir de datos subjetivos esta escala musical ajusta las frecuencias dadas en Hz adaptándose mejor a la percepción humana de los sonidos. La frecuencia en Mels utiliza como referencia la frecuencia de 1000 Hz correspondiente a 40 dB o, lo que es lo mismo, un nivel de ruido asociado a una conversación. La frecuencia de 1000 Hz se equipara a una frecuencia de 1000 Mels, y el resto de los valores se definen de forma logarítmica sobre la escala de frecuencias en Hzs.

2.3. Medio de entrenamiento de datos hacia resultados óptimos (IA)

Los **Algoritmos Genéticos (AGs)** son métodos adaptativos de optimización, búsqueda y aprendizaje inspirados en los procesos de Evolución Natural. Estos algoritmos hacen evolucionar una población de individuos sometiéndola a acciones aleatorias. Los algoritmos genéticos presentan un esquema común que presentamos a continuación:

- ***Generar Población Inicial***, es una función que se encarga de crear al azar la primera población del algoritmo con una cantidad fija de individuos todos distintos entre sí. Lo normal es crear una población inicial con al menos el doble de individuos que de posibles campos a mutar en cada individuo. Es una forma de conseguir suficiente aleatoriedad y evitar el comportamiento de tipo *random walk.*

- ***Evaluar Población***, trata de evaluar todos los individuos de una población para que cada uno adquiera un valor que lo identifique del resto.

- ***Función de selección,*** consiste en la elección de varios individuos de la población para reproducirlos y crear nuevos individuos hijos con parte del genoma de uno y parte de otros. Existen diferentes tipos de selección de entre las que destaca la selección por torneo. Se escogen varios individuos rechazando aquellos que tengan un menor valor de evaluación.

- ***Cruzar Población***, es la parte del algoritmo que se encarga de mezclar los genomas de los individuos obtenidos con la función de selección para crear hijos. Normalmente se crean hijos hasta que la población es dos veces la inicial.

- ***Mutación***, es un cambio al azar que se produce en el genoma de los individuos hijo obtenidos, para que tengan alguna diferencia propia respecto de los padres.

- ***Reemplazo generacional***, escoge de toda la población un número fijo de individuos más reducido. Para la reducción también se suele tener en cuenta el valor de evaluación que se le ha asignado a cada sujeto en el punto anterior.

- ***Criterio de Parada***, es la especificación del momento en el que el algoritmo genético termina ya sea porque ha llegado al umbral establecido, al tiempo o recursos límites para la prueba, o bien ha sido terminado de forma manual por el experto.

3. Representación Lingüística

3.1. Pre proceso de imagen

Muchos de los sistemas de procesamiento de voz, usan el espectrograma de señales de voz como fuente principal de datos (Figura 2). Además, la experiencia en el trato de imágenes del grupo Giara, de la UPNA, y la gran similitud entre un espectrograma de señales de voz con una imagen, ha hecho de él nuestra primera opción de trabajo.

El oído humano es capaz de reconocer con más resolución las señales de frecuencias bajas que las altas. Por este motivo, es normal que el eje vertical del espectrograma se estreche para obtener las frecuencias bajas con más precisión que las altas (el espectrograma estrechado se observa en la Figura 6). Un método muy cómodo para llevar a cabo este proceso es mediante la escala de Mel. Este proceso se conoce como *transformación* y se puede observar en la función que se proporciona con el anexo *grab2mel()*. La aplicación de esta función da como resultado el espectrograma con el eje vertical rojo que se observa en la Figura 5.

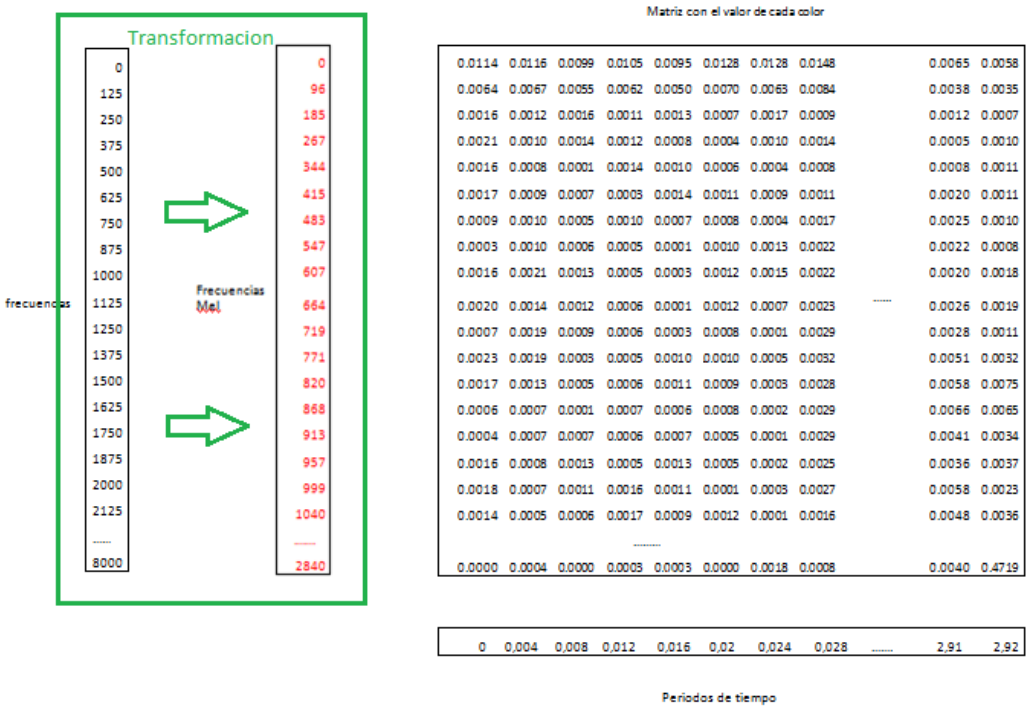


Figura 5. Espectrograma a nivel de datos en el que se muestra qué cambio tiene que realizarse en el eje vertical para pasar de frecuencias en Hzs a frecuencias en Mels. En el anexo hay una función numérica que indica el valor en Mel que corresponde a una frecuencia en Hzs *-freq2mel()-*

Si se reajustan los valores del espectrograma para representarlo con frecuencias de 0 a 8000, se observa de la siguiente manera.

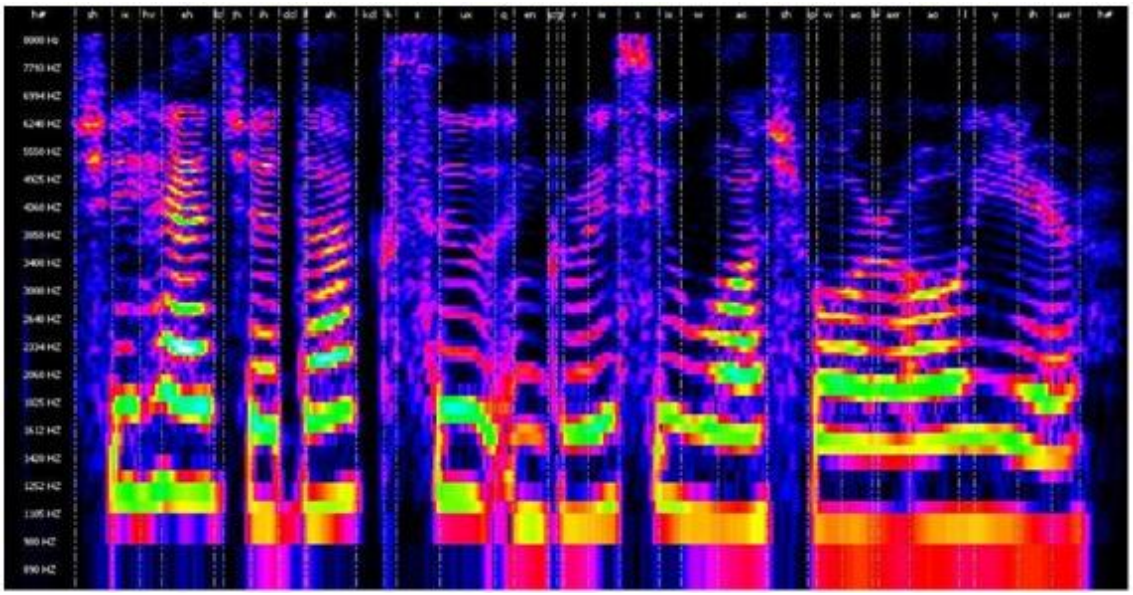


Figura 6. Espectrograma después del estrechamiento por la escala de Mel

Por otra parte, el ser humano no lee todo el espectrograma con total precisión, se centra en unas pocas frecuencias de entre 4 y 6 KHz. Tampoco necesita saber cuál es la amplitud exacta de una onda, ya que con saber si es alta o baja resulta suficiente. Finalmente, no necesita conocer la duración puntual de un sonido, sino su duración relativa (larga, corta, etc.). Por estos motivos, en lugar de realizar los procesos con todo el espectrograma completo, se hace con una reducida y difusa descripción lingüística obtenida del mismo. A este proceso de simplificar la información del espectrograma lo denominaremos *reducción* y puede variar según el caso que se esté tratando como mostramos a continuación.

3.1.1. En el artículo

Después de transformar el espectrograma de frecuencias a la escala Mel, se reduce el eje vertical del espectrograma en 25 bandas. Queremos conseguir un valor para cada una de las bandas y, por tanto, 25 valores para cada muestra en el eje temporal del espectrograma. El representante de cada banda se obtiene ordenando todos los valores que la formaban y haciendo la media sólo de cierta cantidad de valores máximos. Esta cantidad viene definida por una variable llamada MaxN. Siendo N el

número que indica el porcentaje de máximos que se van a tener en cuenta. En el ejemplo del artículo se usa un Max10, y por tanto, se elige un 10% de los máximos. Con esta media se consigue disminuir parte del ruido ya que se obtiene una media que parte de los valores más altos en lugar de un único valor.

Véase el proceso en la imagen siguiente (Figura 7) obtenida, junto los espectrogramas anteriores, del artículo *Recognition of human speech phonemes using a novel fuzzy approach,* de Ramin Halavati, Saeed Bagheri Shouraki y Saman Harati Zadeh. El espectrograma completo queda representado en la Figura 8.

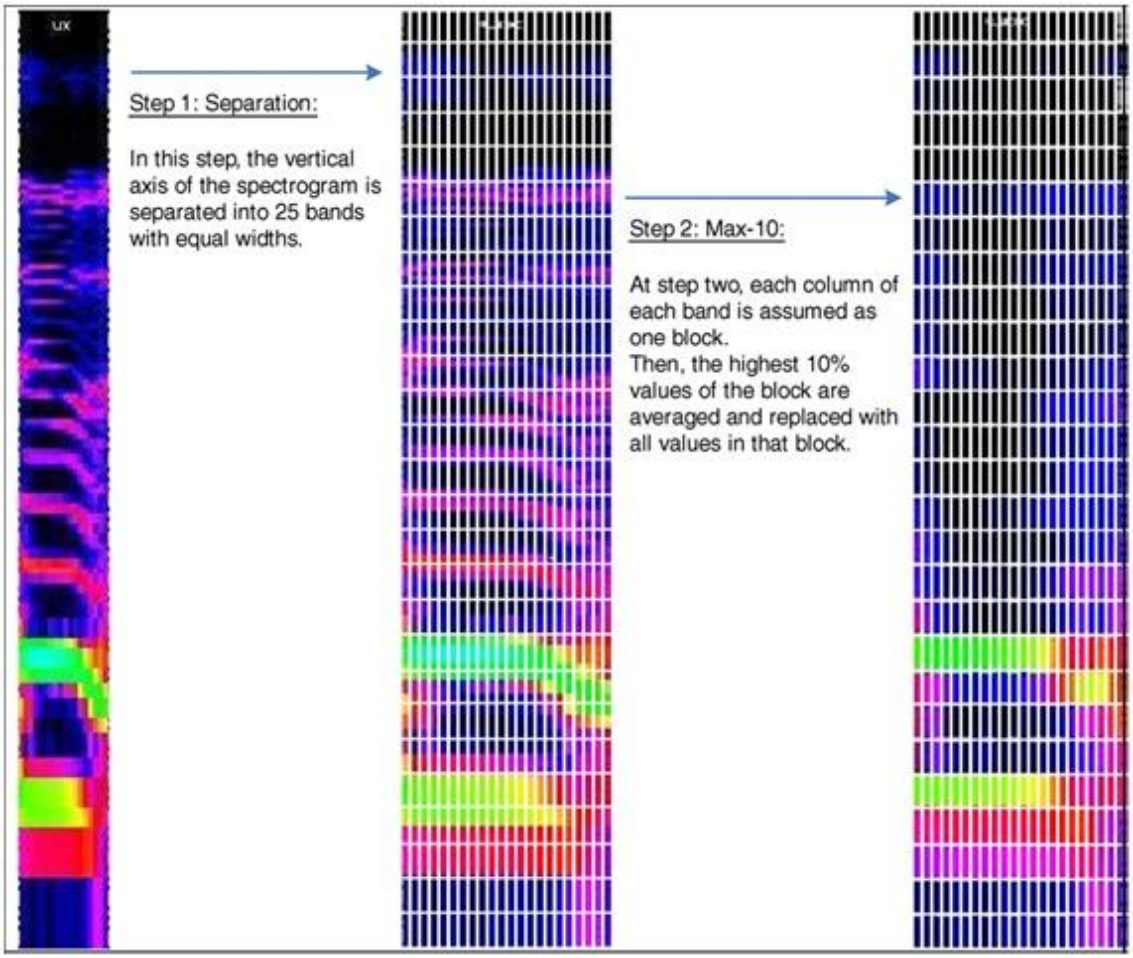


Figura 7. Ejemplo del proceso de reducción.

Nota

Se aprecia que en el espectrograma reducido del ejemplo se obtienen unos valores medios-bajos en lugar de los máximos que deberían observarse según indica el

operador MaxN descrito. Al seleccionar una media de los máximos, en todas las bandas en las que aparece al menos un valor alto debe aparecer un color de intensidad alto, sin embargo, hay bandas con un valor de intensidad muy bajo.

Por este motivo hemos pensado en otras posibles variables de reducción: MinN (mismo mecanismo con los mínimos), Max100-N (en lugar de hacer la media con el 10% se hace la media del 90 %), Min100-N (se hace la media con el 100-N% de los mínimos).

Para el trabajo se ha elegido Min100-N, entendiendo que en el artículo cuando decían Max10 realmente querían hacer la media de todos los valores menos el 10% de los máximos (justo lo contrario). De esta forma se obtiene una buena media, con un espectrograma similar al de la imagen del espectrograma del artículo (Figura 8), y en la que se ha reducido del sonido, el ruido producido por intensidades muy altas.

Una media acotada por ambos extremos, mínimo y máximo, podría ofrecer mejores resultados. No se ha aplicado este cambio en el trabajo para que la comparación entre los tres métodos fuese más equitativa y real. Se puede ver la implementación de este proceso en la función speechreduction() proporcionada en el anexo.

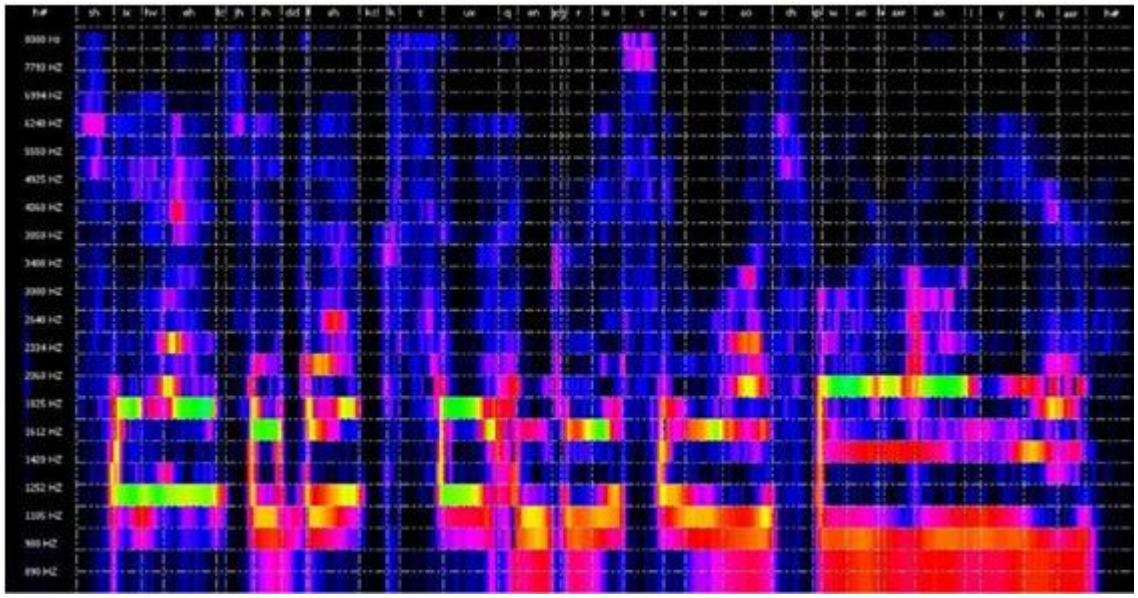


Figura 8. Espectrograma después de la reducción de datos.

3.1.2. En la extensión a intervalos

3.1.2.1. Intervalos en Datos

El método más sencillo para introducir intervalos en el tratamiento del problema es convertir los datos de partida en intervalos, y operar con ellos de modo similar a como se realizaba sobre conjuntos difusos. Para este caso, el proceso de *reducción* tiene que someterse a unos pequeños cambios. Después de transformar el espectrograma de frecuencias a la escala Mel, también reducimos el eje vertical en 25 bandas. El valor que adopta cada banda como representante de la misma deja de ser la media, para tratarse del valor mínimo y del valor máximo de la banda. Así pues, se obtienen dos espectrogramas con la misma estructura que en el artículo de ejemplo, para guardar en uno todos los valores mínimos encontrados, y en otro todos los máximos. Se pueden ver los dos espectrogramas obtenidos del principal con la aplicación MatLab en la Figura 9.

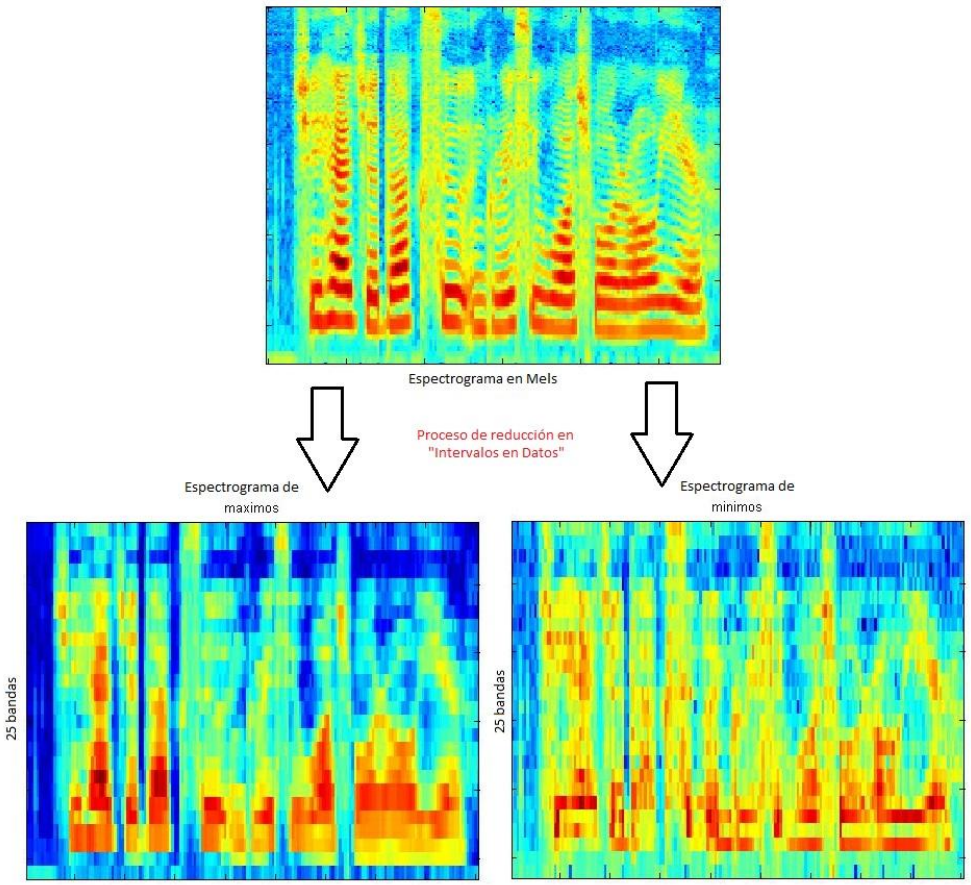


Figura 9. Transformación del espectrograma en frecuencias Mel a los dos espectrogramas máximo y mínimo de 25 bandas cada uno.

Para conseguir esto hay que: ordenar todos los valores de la banda, coger el primero (mínimo) y el último (máximo) de estos valores, y guardarlos, como representantes de la banda, cada uno en un espectrograma distinto, uno de máximos y otro de mínimos (como se ve en la Figura 9).

Se decide trabajar con intervalos porque usar una media como representante de un gran bloque de datos, hace que se pierda mucha información útil. Sin ir más lejos, un fonema largo puede presentar diferentes intensidades para una misma banda, desde muy bajas, hasta muy altas. Con una media se puede obtener un valor central que diste mucho tanto de las intensidades altas que lo definen como de las bajas. Por este motivo es posible que no se obtenga un buen resultado a la hora de reconocer algunos fonemas en una muestra.

Este primer aporte sobre intervalos utiliza los extremos inferior y superior de cada banda. Esta elección se realiza con la t-norma mínimo y t-conorma máximo debido a la sencillez con la que se implementan. Esta implementación puede verse en el anexo en la función *reducciónIntervalos()*.

3.1.2.2. Intervalos en Funciones

Por otro lado, es posible introducir intervalos sobre el problema de una forma más sofisticada, transformando las funciones de pertenencia en funciones intervalo-valoradas y extendiendo los operadores de lógica difusa a operadores capaces de trabajar con intervalos. De esta forma, no se ve modificada la representación de los fonemas respecto al ejemplo del artículo.

En este documento se quieren probar de manera independiente los intervalos en las funciones y en los datos. No se trabaja con ambos métodos a la par para poder comprobar los resultados de cada uno independientemente, valorar sus resultados y elegir los pasos siguientes en función de los más prometedores.

3.2. Representación de fonemas y funciones

Los fonemas a representar, van a seguir la misma estructura en todos los ejemplos y nuevos métodos desarrollados. Al igual que en el espectrograma de señales de voz *transformado* y *reducido*, los fonemas deben comprender 25 bandas. Por el contrario, para cada una de las bandas definidas en los fonemas tiene que haber dos valores. Además, en cada fonema se añade un nuevo valor en la fila 26 relacionado con la longitud del mismo.

Cada fonema se divide en 25 bandas, igual que en cualquier grabación de muestra sobre la que se quiera hacer un reconocimiento (Un ejemplo de un espectrograma de muestra es el representado en la Figura 8). La banda de un fonema guarda información acerca de los posibles valores que puede y debe tener un cuadro de una muestra en esa misma banda (siempre que el cuadro represente parte del fonema). Esta información es el nombre de dos conjuntos difusos (las etiquetas lingüísticas y las

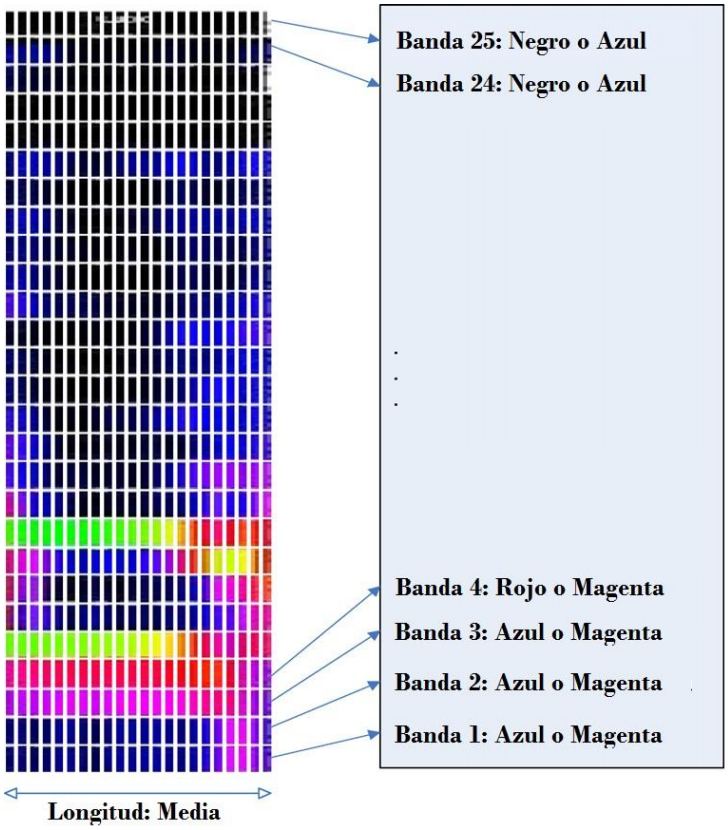


Figura 10. Ejemplo de descripción de un fonema. Cada banda es representada por los dos colores más prometedores. También se aprecia la etiqueta lingüística que representa la longitud.

funciones de pertenencia que definen estos conjuntos se detallan en los subapartados [3.2.1], [3.2.2] y [3.2.3]) y por tanto, el valor de la muestra tiene que pertenecer en un alto grado a alguno de ellos.

En la Figura 10 podemos ver un fragmento de la muestra y la representación de un fonema con las etiquetas que le corresponden en cada banda.

Para terminar de definir un fonema, se le tiene que asignar otra etiqueta lingüística, que determine, en qué conjunto difuso (muy corto, corto, medio, largo y muy largo) se obtiene una mayor pertenencia para un valor de longitud dado. Es lo que en la Figura 10 se denomina *Longitud*.

Se acepta este modelo de representación especificado en el ejemplo del artículo, porque consideramos que es una forma rápida y eficaz de trabajar con los datos. Es rápida por la poca cantidad de información que se usa y eficaz, porque se da una mayor importancia a las frecuencias mejor percibidas por el ser humano.

Nota

En un principio inicializábamos los fonemas a partir de una grabación de muestra con la intención de reducir de forma crítica el espacio de búsqueda, y de facilitar la inicialización aprovechando código ya realizado. Como conocíamos los tiempos en los que empezaba y concluía cada fonema en la muestra, manualmente se indicaban en una función llamada getPhonem(). Esta función calculaba, para cada banda del espectrograma comprendida entre los tiempos en los que empezaba y terminaba el fonema, un valor medio, y para cada valor medio, su grado de pertenencia en cada uno de los conjuntos difusos conocidos con las etiquetas negro, azul, morado, rojo, amarillo, verde, cian y blanco. Cuando se sabía la pertenencia de la media en cada uno de los conjuntos difusos indicados, se ordenaban y se elegían las dos mejores como representación de esa banda del fonema. Como resultado se obtenía una matriz de 25 bandas por dos valores (las dos etiquetas que definían los mejores conjuntos difusos).

Para calcular la pertenencia de un fonema a una longitud, se crea otra función llamada aLongitud(). Si se le pasaba como parámetro el número de intervalos de

tiempo comprendidos entre el inicio y fin del fonema, y la matriz de 25x2 que lo definían, añadía en la matriz obtenida de getPhonem() una nueva fila (26) con una etiqueta lingüística (muy corta, corta, media, larga o muy larga) que indicaba en qué conjunto difuso se había obtenido un mayor grado de pertenencia.

En la práctica, cuando se intentaba obtener una definición más correcta de los fonemas con el algoritmo genético, en muchas pruebas los resultados pronto dejaban de verse modificados (el mejor individuo se cruzaba consigo mismo o uno muy similar, de forma que en pocas generaciones toda la población se había convertido en una réplica). Una de las posibles causas era que la elección inicial de un buen patrón nos llevaba realmente a las cercanías de un mínimo local del que no se podía salir con las funciones de cruce y mutación utilizadas. Por este motivo, decidimos inicializar los fonemas de manera totalmente aleatoria.

Todo este proceso se realiza con las funciones getPhonem() y aLongitud(). La función desde la que se guardan a mano cada uno de los fonemas se llama reduceAPhonemas(), y también se usan para este proceso las funciones cogeMax(), aColores() y generalizada(). Todas están detalladas y explicadas en el anexo que se aporta con este documento.

Las funciones difusas, son las que van a determinar en cuánto, un valor pertenece a un conjunto difuso.

3.2.1. En el artículo

Hay definidas trece funciones con sus cuatro puntos: *inicial*, *primer máximo*, *último máximo* y *final*. Las ocho primeras, hacen referencia a los conjuntos difusos que definen cada banda del fonema. Se representan mediante una escala cromática compuesta por negro, azul, morado, rojo, amarillo, verde, cian y blanco. Las cinco últimas funciones representan valores de longitud, muy corto, corto, longitud media, larga y muy larga. El punto *inicial* de cada función, muestra el valor desde el que un elemento deja de tener pertenencia 0 a ese conjunto. *Primer máximo* y *último máximo* abarcan todo un intervalo en el que la pertenencia es 1. Por último, *final* vuelve a mostrar el punto desde el que la pertenencia vuelve a ser 0. En cualquier caso, si

coinciden *inicial* o *final* con *primer máximo* o *último máximo*, se toma como bueno el valor de pertenencia de los máximos, es decir, 1.

	Inicial	PrimerMax	UltimoMax	Final
Negro	0	0	0.05	0.35
Azul	0	0.05	0.22	0.5
Morado	0	0.23	0.35	0.63
rojo	0.05	0.36	0.5	0.8
Amarillo	0.23	0.51	0.63	0.93
Verde	0.36	0.63	0.8	0.99
Cian	0.51	0.81	0.93	0.99
Blanco	0.63	0.94	0.99	0.99
Muy corto	2	2	3	5
Corto	2	2	6	10
Media	2	6	12	17
Larga	8	12	20	30
Muy larga	12	18	69	99

Los puntos *inicial*, *primer máximo*, *último máximo* y *final* se mueven en un rango [0, 0.99] para las funciones etiquetadas con el nombre de colores, y en el rango [0, 99] para las funciones etiquetadas con el nombre de longitudes. Los valores deben estar ordenados de forma que *inicial<=primer máximo<=último máximo<=final.*

Figura 11. Representación de las funciones de pertenencia para cada una de las etiquetas lingüísticas usadas en la memoria.

Véase la Figura 11 en la que están representadas todas las funciones de pertenencia para este apartado.

3.2.2. En la extensión a intervalos

3.2.2.1. Intervalos en Datos

En el caso de intervalos en los Datos, las funciones no han sufrido ninguna modificación respecto al ejemplo del artículo. En principio, se pensó cambiar las funciones de las longitudes, para que sus rangos fuesen de [0, 0.99] en lugar de [0, 99] y conseguir que estuviese más normalizada. Al no ser relevante en el resultado final, no se realizó ningún cambio.

La única diferencia se aprecia a la hora de aplicar las funciones. Como se dispone de dos conjuntos de datos, hay que aplicar dos veces la misma función, una para el espectrograma de mínimos y otra para el de máximos. De esta forma se obtienen dos valores de pertenencia para una misma función.

Se puede observar cómo funciona este mecanismo en la parte de reconocimiento implementado en la función *jugarEntrenamientoDatos()* y proporcionada en el anexo.

3.2.2.2. Intervalos en Funciones

En el caso de intervalos en las funciones, interesa tener para una misma etiqueta lingüística (Ej.: Negro) ocho puntos en lugar de los cuatro que se definían inicialmente (inicial Max, primer máximo Max, último máximo Max y final Max; inicial Min, primer máximo Min, último máximo Min y final Min). Cuatro puntos tienen que representar la función de pertenencia máxima y otros cuatro la mínima (Figura 12). Debe cumplirse, al igual que las funciones de los otros modelos, que los valores desde *inicial* a *final* estén ordenados por un lado los máximos y por otro los mínimos. Además, también debe cumplirse que los valores *inicial Max*, *primer máximo Max*, *último máximo Max* y *final Max* sean mayores o iguales que *inicial Min*, *primer máximo Min*, *último máximo Min* y *final Min* respectivamente.

	Función de pertenencia Máximos				Función de pertenencia Mínimos			
	Inicial	PrimerMax	UltimoMax	Final	Inicial	PrimerMax	UltimoMax	Final
Negro	0	0	0.05	0.35	0	0	0.05	0.35
Azul	0	0.05	0.22	0.5	0	0.05	0.22	0.5
Morado	0	0.23	0.35	0.63	0	0.23	0.35	0.63
rojo	0.05	0.36	0.5	0.8	0.05	0.36	0.5	0.8
Amarillo	0.23	0.51	0.63	0.93	0.23	0.51	0.63	0.93
Verde	0.36	0.63	0.8	0.99	0.36	0.63	0.8	0.99
Cian	0.51	0.81	0.93	0.99	0.51	0.81	0.93	0.99
Blanco	0.63	0.94	0.99	0.99	0.63	0.94	0.99	0.99
Muy corto	2	2	3	5	2	2	3	5
Corto	2	2	6	10	2	2	6	10
Media	2	6	12	17	2	6	12	17
Larga	8	12	20	30	8	12	20	30
Muy larga	12	18	69	99	12	18	69	99

Figura 12. Definición de las funciones de pertenencia para el ejemplo de Intervalos en Funciones. Vemos que para cada etiqueta lingüística tenemos ocho puntos, cuatro que representan los máximos y otros cuatro que representan los mínimos. Los valores iniciales son los mismos en ambas porque se cambiarán automáticamente en el algoritmo genético.

Para la inicialización se usan dos veces cada función nombrada en los métodos anteriores (Ej.: Negro = NegroMax; NegroMin). Se ha identificado a una como máxima y a otra, con exactamente los mismos valores, como la mínima, y hemos dejado que el algoritmo genético cambie de forma automática los valores que corresponden a una función y otra. El objetivo de esta técnica es reducir la complejidad de las relaciones que habría que mantener con una inicialización aleatoria.

Con el algoritmo genético, los valores que representan las funciones máximas y mínimas se irán ajustando automáticamente, ya que la definición de todas las funciones forma parte del genoma de un individuo, y por tanto, puede sufrir mutaciones.

En este momento hemos expuesto todos los requisitos necesarios para construir y parametrizar el motor de reconocimiento de fonemas que queremos construir. En concreto hemos conseguido:

- definiciones de pertenencia de los colores,
- definiciones de pertenencia de las longitudes, y
- descripción inicial de todos los fonemas.

4. Reconocimiento del habla

En este documento llamamos reconocimiento al proceso mediante el cual, a partir de una muestra de voz, la representación de unos fonemas y la especificación de unas funciones que los definen, se obtiene como resultado el número de fonemas que se han detectado correctamente.

En el ámbito de este trabajo, una *muestra de voz* está representada por el espectrograma de una grabación una vez realizados los procesos de *transformació*n y *reducción*.

4.1. Reconocimiento por bloques

Llamamos reconocimiento por bloques al proceso que se lleva a cabo cuando, para intentar reconocer unos fonemas sobre una muestra, sabemos entre qué momentos (uno de inicio y otro de fin) se pueden encontrar estos fonemas. Toda la información que se encuentra entre cada inicio y fin de la muestra se considera un bloque. En la Figura 13 vemos algunos ejemplos de dónde empieza y dónde acaba alguno de los bloques que tienen la información de un fonema.

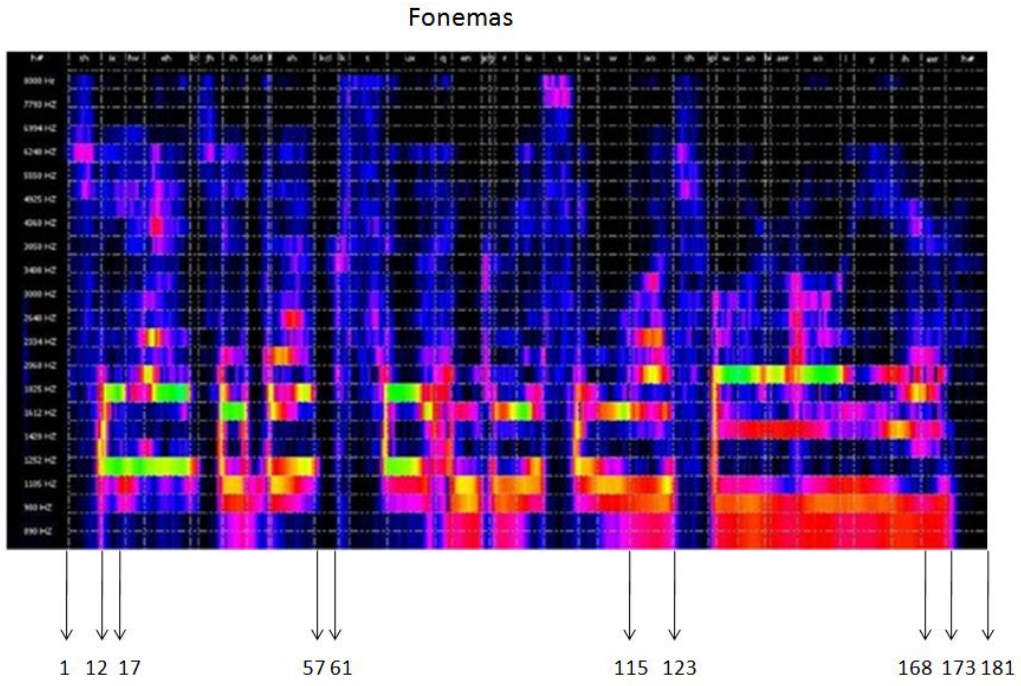


Figura 13. Ejemplo de bloques, con sus límites de inicio y fin conocidos, que contienen la información correspondiente a un fonema. La grabación es un ejemplo que se puede encontrar en una BBDD de voces y en la que también se incluye la información sobre los límites de los fonemas.

El proceso comienza con la delimitación de cada fonema dentro de la grabación (o dicho de otra forma, hay que definir cada bloque manualmente como se observa en la Figura 13, porque conocemos el ejemplo y los fonemas que lo componen). Con la información entre esos intervalos de tiempo, se realiza una comparación con la definición lingüística de nuestros fonemas representados (el proceso de comparación se explica más adelante).

El resultado que se obtiene al final es una lista con todos los fonemas existentes y un *valor de coincidencia* por cada uno. Este valor se mueve en un rango de cero a cien, e indica la probabilidad con la que cada fonema encaja en ese bloque.

Al ordenar esta lista de mayor a menor, siempre estarán en las primeras posiciones los fonemas con mejor *valor de coincidencia*. Cuando se ha cogido el primero en cada uno de los bloques, termina el reconocimiento. El cálculo del *valor de coincidencia* varía ligeramente según el modelo empleado.

4.1.1. En el artículo

El ***primer paso*** consiste en comprobar, como ya se ha indicado en la representación de fonemas [3], en qué medida un punto de la muestra pertenece a los colores que definen un fonema. Llamamos punto de la muestra a un valor del espectrograma, en una banda e intervalo de tiempo concreto. Como el fonema está definido con dos colores por banda, para cada punto se obtienen dos valores de pertenencia. Todos los puntos de un mismo instante se tratan a la vez. En cada banda, se escoge el máximo de los dos valores de pertenencia, y de los 25 valores máximos del instante de tiempo completo se selecciona el mínimo. Esto se realiza para todos los fonemas existentes. Al final, para cada instante habremos guardado un valor de pertenencia para cada fonema. Una explicación gráfica de este proceso se encuentra en la Figura 14.

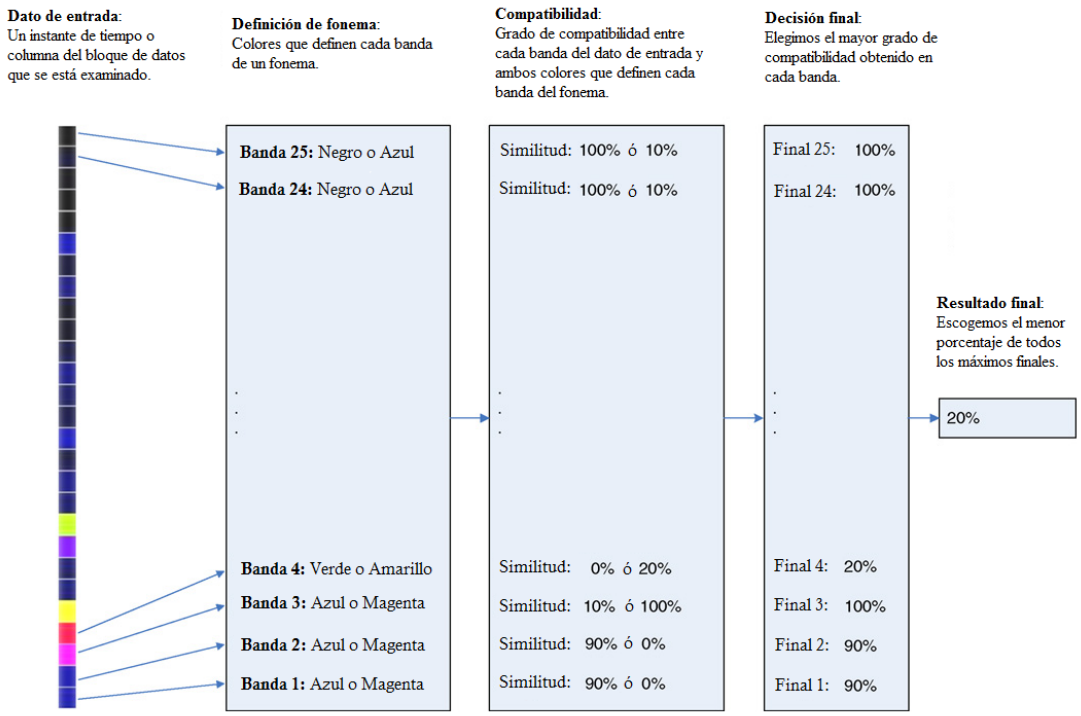


Figura 14. Ejemplo de realización del *primer paso* con un instante de tiempo de un bloque y una definición de un solo fonema.

El ***segundo paso,*** empieza cuando ya se ha realizado el *primero* sobre todos los periodos de tiempo que comprendían el bloque. Con tantos valores mínimos por fonema como periodos de tiempo en el bloque, se define un nuevo operador llamado AverageN, siendo N un número que indica el porcentaje de valores que no se van a despreciar al hacer la media. Para N igual a 80, y X el número intervalos del bloque examinado, el valor que se va a guardar como final para cada fonema, es una media desde 1 hasta (80*X)/100 de los valores que se han guardado en el *primer paso*, ordenados de menor a mayor. Como queremos que el resultado se devuelva entre 0 y 100, y los valores con los que se hace la media están comprendidos entre 0 y 1, multiplicamos por 100 el resultado.

El ***tercer*** y último ***paso*** consiste en, con el tamaño del bloque, obtener un valor que indique en qué medida, ese tamaño pertenece a la longitud que define a cada fonema. En este momento se está trabajando con un bloque de entrada de longitud determinada y con una etiqueta lingüística asociada a cada fonema para representar su longitud. Para probar si el tamaño del bloque se corresponde con el tamaño asociado a

cada fonema, se calcula el valor de la longitud del bloque en la función de pertenencia asociada a la etiqueta lingüística de cada fonema. Si se multiplica este valor por el dato obtenido con AverageN, se obtiene como resultado el *valor de coincidencia*.

Este *tercer paso*, tiene como fin ajustar la media obtenida en el *segundo paso*. Como de la longitud se obtiene un valor entre 0 y 1, al multiplicarlo por la media del apartado dos se consigue un valor, el mismo si la longitud encaja, o más bajo si la longitud no encaja perfectamente (porque se multiplica por un valor menor que 1).

El código y explicaciones de la función que realiza estos pasos se encuentra en el anexo con el nombre de *jugarEntrenamiento()*.

4.1.2. En extensión a intervalos

4.1.2.1. Intervalos en Datos

Esta primera y más sencilla implementación de intervalos difiere del ejemplar del artículo en que precisa de dos muestras con las que trabajar, una muestra con valores mínimos y otra con máximos (Como se indica en [3.1.2.1]). Por este motivo, el ***primer*** y ***segundo paso*** no varían excepto por el hecho de que se realizan dos veces una sobre cada muestra. De esta forma se obtienen dos conjuntos de datos con los valores calculados de la muestra de mínimos y los calculados de la muestra de máximos.

La primera parte del ***tercer paso*** (obtener un valor que indique en qué medida el tamaño del bloque pertenece a la longitud que define a cada fonema) es exactamente igual, ya que al ser la longitud del bloque idéntica en ambas muestras no hace falta calcular su pertenencia dos veces. La segunda parte, por el contrario, si ha tenido que cambiarse porque el resultado final obtenido para cada fonema en el *segundo paso* son dos, sus valores calculados de la muestra de mínimos y los calculados de la muestra de máximos.

En el apartado anterior [4.1.1, tercer paso], la media se multiplicaba por el valor que se obtenía de la longitud del bloque. Aprovechando los dos valores que proporcionan los intervalos, se usa el grado de pertenencia que se obtiene de la longitud

(número entre 0 y 1) para conseguir una única media representativa por medio de la función:

mediaMínimos + (PertenenciaLongitud · (mediaMáximos – mediaMínimos)).

De esta forma se consigue que, dependiendo del grado de pertenencia calculado con la longitud del bloque, se aproxime el intervalo hacia su valor mínimo o máximo.

Con este método se obtienen valores más ajustados porque en lugar de asignar un cero a un fonema cuando no encaja su longitud, se le asigna un valor mínimo que está más cerca de la coincidencia real del fonema en la muestra.

La función que incorpora el código de estas explicaciones con sus respectivos comentarios se llama *jugarIntervalosDatos().*

4.1.2.2. Intervalos en Funciones

Cuando se trata de intervalos en las funciones, los tres pasos a seguir son similares a los anteriores.

El ***primer paso*** consiste en comprobar, al igual que en los otros apartados, en qué medida un punto de la muestra pertenece a los colores que definen un fonema. La diferencia está en que ahora un color está definido por dos funciones de pertenencia [3.2.2.2] y como cada fonema está definido por dos colores por banda, se obtienen cuatro valores de pertenencia (un intervalo para cada color, es decir, dos valores para cada color). Así como antes se elegía el mayor valor como resultado final de cada banda, ahora se coge el intervalo con el mayor mínimo de los intervalos. Cuando se ha obtenido un intervalo para cada una de las bandas que componen un instante de tiempo del bloque, se selecciona el mínimo de los valores mínimos de todos los intervalos y el mínimo de los valores máximos. Al realizarse este paso sobre cada uno de los instantes de tiempo que componen el bloque, se obtienen dos conjuntos de datos con todos estos valores igual que en [4.1.2.1]. Uno con los valores calculados a partir de los mínimos de los intervalos y otro con valores calculados a partir de los máximos.

El ***segundo paso*** es igual que en intervalos con datos, se realiza el *segundo paso* del ejemplo del artículo para cada uno de los dos conjuntos creados (conjunto con valores mínimos y conjunto con valores máximos obtenidos en *primer paso*).

El ***tercer paso*** es similar al de intervalos en datos, ya que llegados a este punto, las estructuras de datos con las que se trabaja son las mismas (a nivel de implementación, dos matrices de "numero de fonemas x longitud de bloque"). La diferencia está en que ahora hay una función de máximos y otra de mínimos para cada tipo de longitud, por lo que al calcular la pertenencia de la longitud del bloque en los conjuntos que definen las funciones de longitud, se obtienen dos valores.

El proceso por el que se obtiene un resultado final, consiste en multiplicar el valor de pertenencia a la función máxima de la longitud del bloque, por el valor máximo del intervalo obtenido en el *segundo paso*, y el valor de pertenencia a la función mínima de la longitud del bloque, por el valor mínimo del intervalo. De esta forma se obtiene un intervalo como primer resultado final, pero se escoge la media, como el *valor de coincidencia* representativo del fonema, ya que es una forma sencilla de tener en cuenta ambos valores mínimo y máximo de cara a mostrar un resultado.

Observamos la implementación de estos pasos en la función que se aporta con el anexo *jugarIntervalosFunciones().*

4.2. Reconocimiento simultáneo

Otra estrategia de reconocimiento más enfocada al uso en la vida cotidiana es el reconocimiento simultáneo. Consiste en analizar todos los cuadros (un cuadro es el conjunto de valores que hay en un determinado instante de tiempo) de forma continua según se va recibiendo un sonido.

Nota

En principio, se intentó realizar el trabajo pensando en un tipo de reconocimiento simultáneo, pero por la cantidad de tiempo necesario y el estudio necesario de otros factores adicionales como: identificar cuando un fonema es bueno,

cómo leer el sonido de forma interactiva, evitar leer fonemas de longitud muy corta como buenos cuando hay fonemas largos que encajan, etc., sólo se desarrolló e implementó una sencilla simulación para comprobar su funcionamiento y preparar las bases para un futuro proyecto.

Entendemos como reconocimiento de sonido de forma continua, a reconocer lo que se dice mientras se recibe el sonido. El trabajo de base será el mismo que en el caso de reconocimiento de bloques, por lo que esta sección se limita a explicar la implementación desarrollada inicialmente para este proceso.

Se va a tomar como datos de partida una grabación completa transformada y reducida ya que resulta indispensable para realizar validaciones medibles.

A partir del primer índice conocido (normalmente el 1; definido en una tabla -*tiempos*-), se recorre uno a uno todos los cuadros de la muestra. Para cada cuadro se realiza una comparación con todos los fonemas como se explica en el *primer paso* del reconocimiento por bloques (punto [4.1], cada método a su manera).

A nivel de implementación, cada vez que se realiza el *primer paso* sobre cada cuadro, se calcula un valor que corresponde a cada fonema, en el caso del artículo, o un intervalo, en los otros dos casos. Estos valores resultantes se almacenan en una matriz (*minValues*) o dos (*minValueMin*, *minValueMax)* dependiendo de la situación. Las filas de la matriz representan fonemas y las columnas los cuadros examinados.

Se puede ver en la Figura 15 un ejemplo del proceso mencionado (para el caso del artículo).

Antes de repetir el proceso con el siguiente cuadro, se comprueba si el valor calculado en el *primer paso* para cada fonema, es mayor que cierto umbral (*minParaContar*) definido al comienzo del algoritmo (hay una tabla que lleva la cuenta de valores consecutivos mayores que el umbral de cada fonema –*longitudes*-). De ser así, se aumenta en uno el valor de la tabla *longitudes* de aquellos fonemas que cumplen este criterio. De lo contrario, se reinicializa el valor del fonema en la tabla *longitudes* a cero.

Después de incrementar *longitudes*, para todos aquellos fonemas que tienen al menos un valor mayor que el umbral (el último calculado), llevamos a cabo el *segundo paso* explicado en el reconocimiento por bloques. Esta vez la media no se realiza con la variable *AverageN* sino con la última sucesión de valores consecutivos, mayores que el umbral, que tiene un fonema.

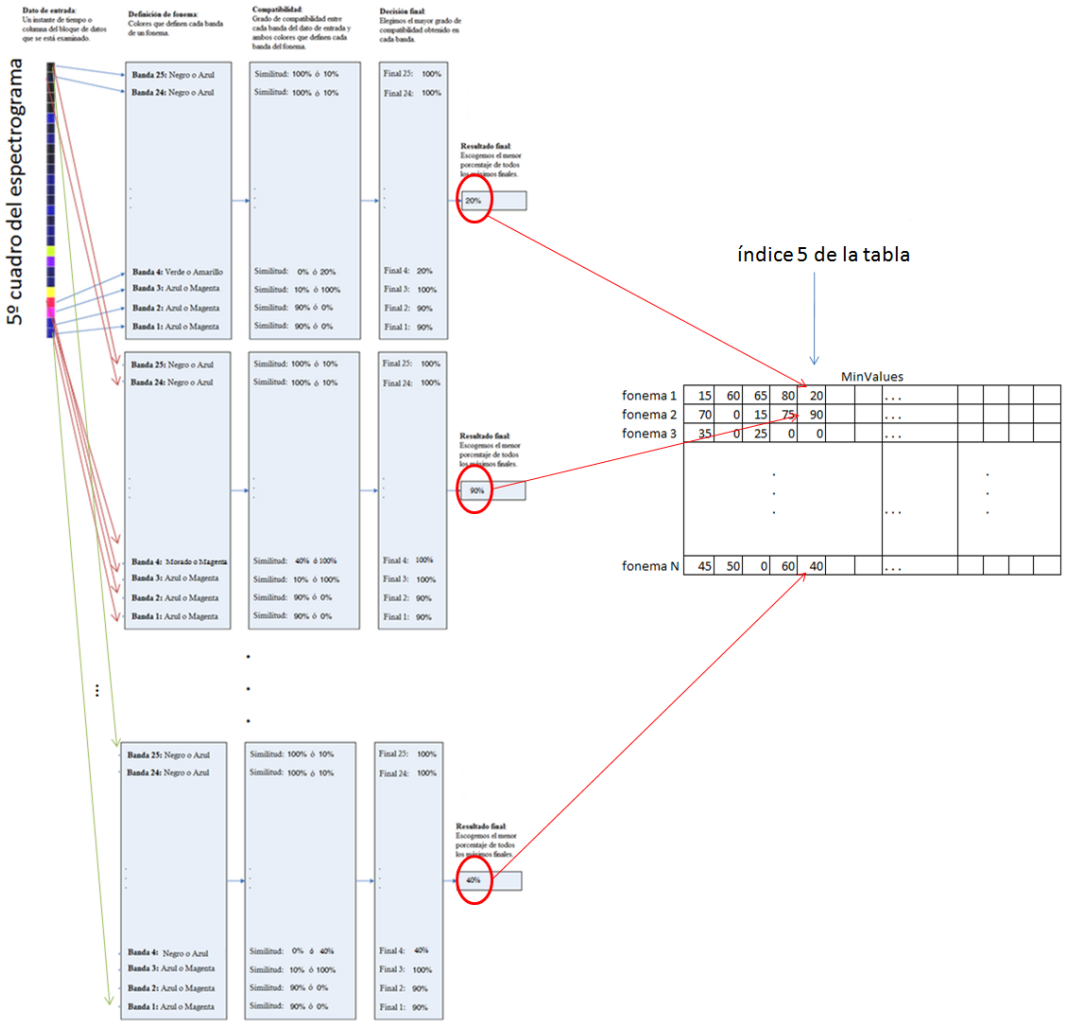


Figura 15. Ejemplo de reconocimiento y guardado de los porcentajes mínimos compatibles entre el cuadro y la definición de los fonemas, para el ejemplo del artículo.

El *tercer paso* es igual que el explicado anteriormente en cada uno de los métodos. La diferencia se encuentra en que esta vez no se tiene una longitud de bloque para ajustar la media obtenida en el *segundo paso*. La longitud cuyo grado de

pertenencia va a ajustar esta media es la que tiene cada fonema en la variable *longitudes.*

Finalmente, si alguno de estos últimos resultados calculados en el *tercer paso* para cada cuadro, es mayor o igual que otro umbral establecido al principio del algoritmo (*minParaValidar),* la función finaliza y devuelve el fonema correspondiente para que sea guardado en una lista de fonemas encontrados. Tras esto, la función vuelve a empezar otra vez a partir del siguiente índice conocido de la tabla *tiempos*. El orden final de estos fonemas en la lista indica el orden en el que los fonemas se han encontrado a lo largo de la grabación.

Se considera que un buen valor para *minParaContar* es 60 porque implica que todos los valores que se encuentran consecutivos coinciden como mínimo en más de un 50% con cada cuadro de la grabación. A *minParaValidar* se le asigna 85 porque se quieren descartar aquellos fonemas en los que haya por lo menos un valor pequeño que baje la media, ya que estos valores pequeños son los que pueden diferenciar un fonema de otro.

En caso de los intervalos, como se tiene en cuenta varios valores, *minParaContar* debe comprender un intervalo. En este caso tomamos la medida del caso inicial (60) como el valor mínimo a partir del cual se tendrá en cuenta un intervalo, y 100 como el máximo.

El umbral m*inParaValidar* en el caso de Intervalos en Datos no varía porque por medio de la función: *min+(numero*(max-min))*, se obtiene un único valor final que se compara con el umbral.

Para Intervalos en Funciones tampoco es necesario ya que, a pesar de trabajar todo el rato con dos valores y obtenerse dos resultados finales (uno mínimo y uno máximo), se ha realizado la media de ambos para poder devolver, en última instancia, un único valor que represente, de forma más sencilla, la coincidencia del fonema con la muestra.

Nota

Los valores que se obtienen y se comparan con los umbrales se mueven en un rango de 0 a 100, por lo que se puede considerar que se trata de porcentajes.

Todo este proceso se puede observar con la misma función que la de reconocimiento por bloques si en lugar de indicar inicio y fin en el paso de parámetros, se indica el inicio, y en final se pone un 0.

Nota

Indicamos "inicio" en el reconocimiento simultáneo porque nuestro ejemplo está implementado para controlar el número de fonemas que se detectan. Si hacemos que "inicio" sea el inicio de cada fonema dentro de la grabación, como hacíamos en bloques, podemos llamar a este proceso para que busque, desde cada inicio, un fonema. En el momento en el que encuentre alguno, el programa se detiene y empieza nuevamente desde otro "inicio" que se le indica. De esta forma controlamos el número de fonemas reconocidos ya que, en cualquier caso, se va a buscar cada fonema desde donde se supone que debería empezar.

La función reconocimiento(), utilizada en el reconocimiento por bloques, es la que ejecuta las funciones jugarEntrenamiento(), jugarIntervalosDatos() o jugarIntervalosFunciones() (dependiendo del método que se utilice) indicándoles el índice desde el que tienen que buscar un fonema. Cuando se encuentra algún fonema, reconocimiento() se encarga de guardarlo y re llamar a estas funciones con el siguiente índice de la tabla –tiempos-. Para el uso de reconocimiento simultáneo, en reconocimiento() hay que cambiar a cero el último argumento que correspondería con el índice "final" .

5. Entrenamiento

En esta sección, tratamos de obtener la mejor parametrización de los fonemas y funciones por medio de un algoritmo genético. Este proceso se realiza sobre grabaciones en las que expertos humanos han clasificado tanto los fonemas de las grabaciones como su duración exacta, ya que para ajustar los fonemas y funciones, así como para poder evaluar cada población, hay que realizar un aprendizaje supervisado.

Nota

Se necesitan los datos clasificados porque el hecho de conocer entre qué puntos se encuentra un fonema, hace que no se tenga que mirar en la muestra completa cada vez que se busca uno. Mirar la coincidencia de todos los fonemas en un bloque de datos de una muestra conlleva mucho menos esfuerzo que mirar todos los fonemas en una muestra completa. Además, para poder saber cuándo se ha terminado de examinar una muestra, es preciso conocer el número de fonemas que había en ella.

Como una población está formada por muchos individuos y un individuo por una gran cantidad de datos, es de vital importancia aumentar la eficiencia del algoritmo genético en la medida de lo posible. Conocer estos datos clasificados reduce el tiempo que se tarda en evaluar una población y por tanto la mejora.

A continuación se detallan los pasos generales de un algoritmo genético con los conceptos comunes a todos los métodos explicados (Intervalos en Datos, Intervalos en Funciones y método del artículo):

- **Generar Población Inicial**: Por medio de la función *rand()* de MATLAB, creamos de forma aleatoria la descripción de los fonemas de cada individuo de la población. La definición inicial de las funciones no se realiza al azar porque ya tienen asignada una inicialización (Figuras 11 y 12).

Nota

En primer lugar y partiendo de la definición de fonemas que nombramos en el apartado [3.2], se creaba la población inicial aplicando una cantidad de mutaciones sobre la definición de fonemas existente. Al notar que en muchas ocasiones los resultados llegaban a un mínimo local del que resultaba imposible salir, se dedujo que la inicialización calculada de los fonemas se encontraba muy próxima a este mínimo local (se explica el porqué en el tercer párrafo de la nota del apartado [3.2]). Por este motivo se decide descartar esta inicialización y crear la población inicial de forma totalmente aleatoria.

Debido a que cada individuo de la población está compuesto por toda una definición de los fonemas existentes y por toda la definición de las funciones de pertenencia, no se han generado tantos individuos por población como se debería para asegurar la aleatoriedad (doble de individuos que genes de un individuo; si un individuo tiene 10 genes, la población debería ser de 20 individuos), ya que por la cantidad de datos con los que se trabaja, se tardaría mucho tiempo en obtener algún resultado. Las poblaciones se han probado con entre 40 y 200 individuos.

La creación de la población inicial se puede observar en la función *generarPoblacion()* proporcionada con el anexo.

- **Evaluar Población**: La función de evaluación se usa tanto para evaluar la población inicial como para cualquier otra población a lo largo del entrenamiento. Un individuo se evalúa intentando reconocer en una grabación de muestra transformada y reducida (que tiene como base el algoritmo genético), los fonemas que la componen. La descripción de estos fonemas junto las funciones que los definen se encuentran en el genoma del individuo.

El proceso de reconocimiento es el mismo que el explicado en el apartado [4.1] a excepción de que ahora, con la lista de valores de coincidencia que se obtiene en cada bloque de la muestra, se busca el fonema que representa el bloque y se suma al resultado final (evaluación final del individuo): un 1 x *valor de coincidencia* si se

encuentra en la primera posición de la lista, un 0.5 x *valor de coincidencia* si se encuentra en la segunda, un 0.4 x *valor de coincidencia* si está en la tercera o la cuarta, un 0.3 x *valor de coincidencia* si está entre la quinta y la séptima y un 0.15 x *valor de coincidencia* si lo localizamos entre la octava y la onceava.

Como el *valor de coincidencia* es el porcentaje de similitud del fonema con la muestra, se consigue, no sólo tener en cuenta la posición en la que el fonema correcto se ha encontrado en la muestra, sino el porcentaje de parecido con la misma.

Nota

Si un fonema pertenece a un bloque con un 20% de parecido y el resto de fonemas no superan este porcentaje, al ser el primero de la lista se sumaría un 1 a la variable –evaluación- pero no se habría obtenido un buen resultado real de coincidencia. Sin embargo, si se tiene en cuenta además de la posición, el porcentaje, se estaría sumando un 0.2 y podríamos distinguir, de entre los diferentes individuos que tienen el fonema en la primera posición de la lista, aquel con mejor valor de coincidencia.

La función que realiza todo este proceso se denomina *evaluarPoblación()*

- **Función de selección:** Se encarga de seleccionar dos individuos de toda la población por medio de un torneo. Se eligen 2 posibles individuos como primer padre al azar eliminando el que tiene menor evaluación. Se vuelve a hacer lo mismo para el segundo padre. Para evitar la selección repetida de dos individuos a priori, al elegir el segundo padre se comprueba que no es el mismo que el primero. De ser el mismo, se vuelve a elegir por torneo entre dos otro. Si al tercer intento el azar ha seleccionado al mismo padre de nuevo, se toma como bueno.

Nota

La intención era hacer un torneo entre 3 para cada padre, pero al ser tan reducida la población se prefirió un torneo entre 2 para que hubiese más variedad en el cruce (Con una población de 20-40 y un torneo de 3, casi siempre se obtenían los mismos padres buenos y la población no variaba).

La este mecanismo de elección de los progenitores se puede observar en el contenido de la función *cruzarPoblación().*

- **Cruzar Población:** Se encarga de mezclar el genoma de dos individuos seleccionados (los dos individuos no eligen nunca un componente del mismo sitio. Ejemplo: la definición de la función Negro) para crear de la unión varios hijos.

La función que lleva a cabo el cruce de los individuos se proporciona con el anexo bajo el nombre de *cruzar().*

- **Reemplazo generacional:** Realiza la selección de los mejores individuos de la población reduciéndola al número de individuos de la población inicial (en este punto la población es el doble que la inicial porque tiene tanto padres como hijos).

Se proporciona la implementación de este proceso en la función *nuevaGeneración().*

- **Criterio de Parada:** Se llevan a cabo los procesos de *reemplazo generacional*, e*valuar nuevos individuos* y *cruzar población* hasta que se encuentra algún individuo cuyo valor de evaluación sea igual o mayor a un umbral establecido. En nuestro caso el umbral es cuando se ha encontrado correctamente al menos un 90 % de los fonemas de

la grabación (si disponemos de 36 fonemas, cuando la función de evaluación es 32 termina el algoritmo genético).

Nota

Con este método se está evitando mejorar en más de un 90 % los fonemas, por lo que en siguientes versiones, se ha pensado cambiar por un sistema de control de iteraciones (el algoritmo genético termina cuando se han realizado un número X de iteraciones sin cambios en el genoma).

Algo muy característico de todo algoritmo genético es la mutación, ya que es la que va a permitir que los elementos de una población cambien y se ajusten automáticamente hacia el resultado deseado.

- Una ***mutación*** es un cambio que se puede producir de forma aleatoria en un genoma de un individuo. La mutación es propia de cada sistema de entrenamiento. Si nos centramos en nuestro caso, tiene la propiedad de cambiar tanto un valor de alguna de las funciones de pertenencia como un color de alguna de las bandas que componen algún fonema.

En las funciones de pertenencia que representan colores, se puede sumar o restar 0.01 a cualquiera de los puntos siempre que no se obtenga un valor que sea menor que 0 ó mayor que 0.99. En el caso de longitudes, como el rango es de 0 a 99 se podrá sumar o restar 1 siempre que no se obtengan valores negativos o mayores que 99. En cualquiera de los casos, hay que tener en cuenta que los valores que definen cada una de las funciones de pertenencia tienen que estar ordenados de menor a mayor como ya hemos indicado en el apartado [3.2].

Los colores que definen cada banda de un fonema también pueden verse alterados. Un color, de alguna de las bandas de un fonema, puede cambiarse por otro que esté inmediatamente después o antes en la definición de las funciones de pertenencia. En el caso de que una banda que estuviese representada por *verde* mutase, podría cambiar a *amarillo* o *cian* (Véase figura). Con las longitudes pasa exactamente lo mismo.

Dentro de las funciones localizadas en los bordes, y aunque carezca de sentido teórico (porque no tiene sentido que un fonema caracterizado por una longitud muy larga pase de repente a tener la longitud más corta y viceversa), queremos que se pueda pasar de negro a blanco, de muy corto a muy largo y viceversa, para mantener una distribución de probabilidad uniforme para la mutación. De lo contrario al final todos los valores acabarían aproximándose a un extremo o a otro.

Esta función se llama *mutar()* y se encuentra en el anexo proporcionado con la memoria.

5.1. En el artículo

- **Cruzar Población**

El **cruce de poblaciones** se basa en el azar. Se escogen partes completas de la definición de cada individuo seleccionado (una función de pertenencia completa Ejemplo: Azul 0 0.05 0.22 0.50 ó una definición de un fonema completo como en la Figura 11).

Dados los individuos *padre* y *madre*, si *padre* cede a un hijo la definición de la función de pertenencia amarillo, *madre* deberá ceder su definición de amarillo a otro hijo. Se ve pues, que de un padre y una madre se obtienen dos hijos completamente diferentes, ya que lo que uno hereda del padre el otro no lo tiene y viceversa.

- **Evaluar Población**

La **evaluación de la población** realiza internamente el proceso de reconocimiento, con cada uno de los individuos que forman la población. Existe la función *evaluarPoblacion()* que llama a *jugarEntrenamiento()* tantas veces como bloques hay en la muestra, para obtener por cada bloque, una lista de valores de coincidencia de cada fonema y poder calcular la evaluación del individuo como ya se ha explicado en [5, Evaluar población inicial].

5.2. En extensión a intervalos

5.2.1. Intervalos en Datos

En el entrenamiento de la versión con intervalos en los datos (que recordamos se trabaja con dos muestras, una con los valores mínimos y otra con los máximos, ambos obtenidos del espectrograma de la muestra inicial transformado, al hacer la reducción), los procesos del programa principal son exactamente los mismos que los creados en el ejemplo del artículo.

- **Cruzar Población**

El **cruce de poblaciones** en Intervalos en Datos es el mismo que en el ejemplo del artículo, ya que el intervalo se encuentra en la muestra de voz y no en la descripción (fonemas y funciones) de cada individuo.

El genoma de un individuo está compuesto, al igual que en artículo, por una definición de todas las funciones de pertenencia (Figura 11) y por la descripción de todos los fonemas (25 bandas, con 2 colores en cada banda, y una 26 para la longitud). Para cruzar un individuo habría que seleccionar o definiciones enteras de funciones (Ejempo: Negro 0 0 0.5 0.35) o descripciones de fonemas enteras (las 25 bandas con sus dos colores y la longitud).

- **Evaluar Población**

En Intervalos en Datos el reconocimiento se realiza de forma distinta a los otros dos métodos, por medio de la función *jugarIntervalosDatos()*. La **función de evaluación** (*evaluarPoblación()*) se modifica para que llame a esta función.

El reconocimiento con Intervalos en Datos necesita de dos muestras sobre las que trabajar. En el cuerpo principal del programa se tiene que transformar la muestra de voz inicial en dos muestras tal y como se define en el apartado [3.1.2.1].

Esta función de transformación se llama *reducciónIntervalos()*.

5.2.2. Intervalos en Funciones

Nuestro segundo ejemplo de intervalos tiene una mayor variedad de cambios, respecto del ejemplo del artículo, en el proceso de entrenamiento. Esto se debe a que en el genoma de un individuo se ha tenido que insertar otra definición de funciones de pertenencia. Un genoma está compuesto por las funciones de pertenencia iniciales (que hacen de funciones de pertenencia máximas), las nuevas añadidas (que hacen de las funciones de pertenencia mínimas) y la descripción completa de todos los fonemas (25 bandas con dos colores y su longitud).

- **Cruzar Población**

El cambio a realizar respecto del modelo del artículo, consiste en modificar la función **cruzar población** para que, al seleccionar de un padre la definición de una función de pertenencia, se cojan los ocho puntos que la forman en lugar de cuatro (viendo la Figura 12, un ejemplo sería coger tanto Azul Máximos como Azul Mínimos).

- **Evaluar Población**

Como se ha indicado anteriormente, se cambia la función a la que llama *evaluarPoblacion()* por *jugarIntervalosFunciones()*. Además, las funciones de pertenencia que usa el algoritmo genético para reconocer los fonemas en la muestra y que tiene definidas al principio del cuerpo principal, hay que ampliarlas a 8 puntos como se indica en [3.2.2.2] (las funciones de pertenencia están definidas en el cuerpo principal de las funciones que hacen de algoritmo genético en cada uno de los tres métodos vistos –*algoritmoGenetico()*-).

- **Mutación**

Como la definición de las funciones de pertenencia es distinta a la de los otros dos métodos anteriores (véanse Figura 11 y Figura 12), hay que adaptar la **función de mutación** para que se puedan mutar ocho valores por función de pertenencia en lugar de

cuatro. Además, en caso de producirse una mutación tenemos que comprobar que se cumplen todas las prioridades de orden mayor-menor tanto entre la definición de pertenencia de los mínimos, como la de los máximos, como entre ambas (explicadas en el apartado [3.2.2.2]).

Nota

En todos los ejemplos se ha creado un sistema de seguridad para que, cada cierto número de poblaciones examinadas, guarde el mejor individuo y sus funciones de pertenencia.

6. Experimentos

En el apartado de experimentos se intenta comprobar la eficiencia de los métodos utilizados. Todos los experimentos se han llevado a cabo con una misma duración de 3 horas para poder obtener comparaciones reales de los resultados.

Nota

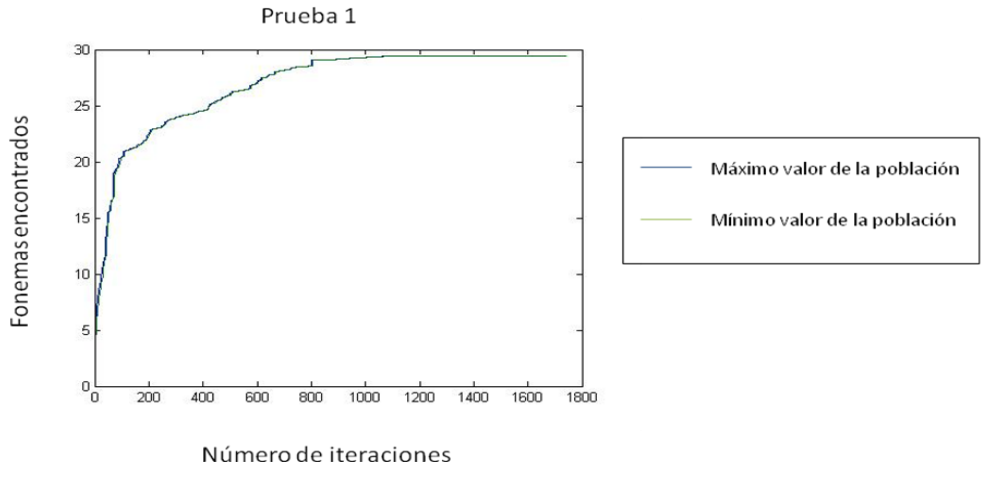
No podemos identificar a nivel de reconocimiento qué método es mejor, porque sólo hemos podido trabajar con una muestra de voz libre que se proporciona en la "Linguistic Data Consortium" (LDC). Los experimentos llevados a cabo, sirven para demostrar que en intervalos, a pesar de trabajar casi con el doble de información y realizarse un menor número de iteraciones, en cada entrenamiento se siguen obteniendo una gran parte de los fonemas de la muestra con mucha precisión.

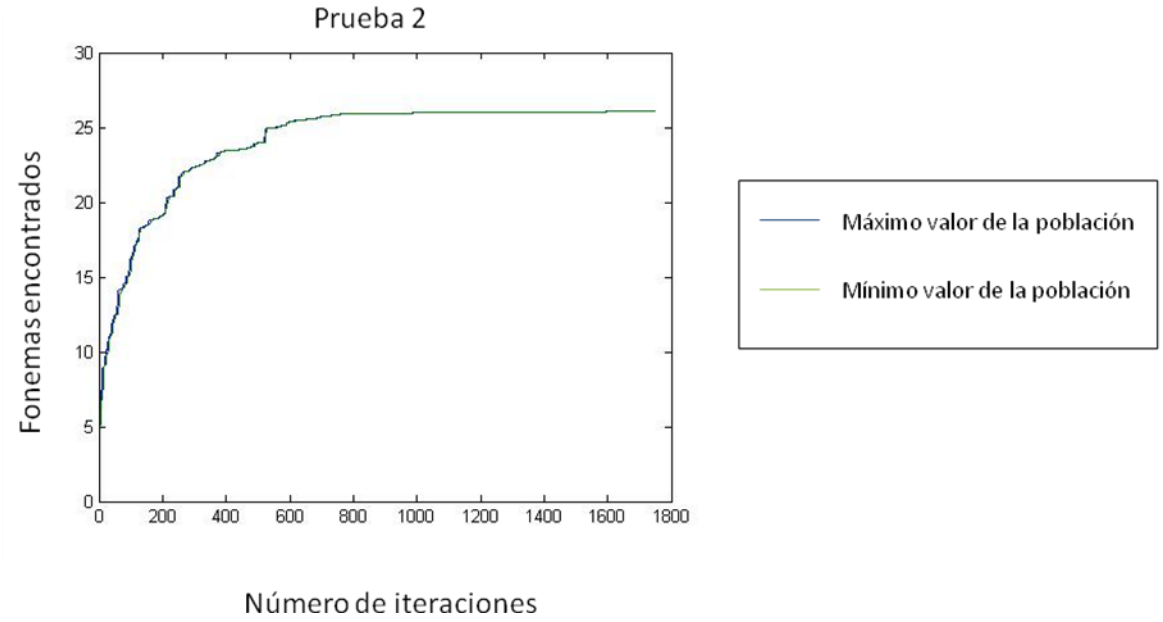
Al trabajar con diferentes muestras de voz, sería probable que ambos métodos intervalares diesen mejores resultados, ya que tienen en cuenta mucha más información (toda la que se encuentra dentro de cada intervalo) que el método del artículo y es posible que abarque un mayor número de tonalidades.

6.1. Entrenamiento con 40 individuos por población

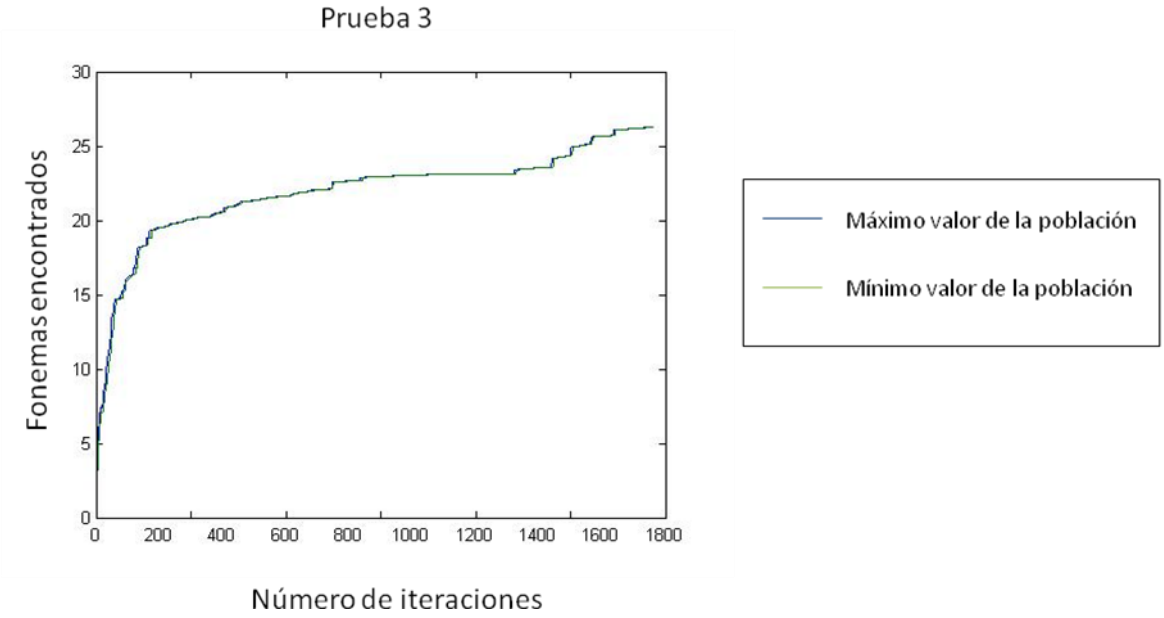
La prueba consiste en realizar un entrenamiento con 40 individuos por población para ver cómo evoluciona el reconocimiento de los fonemas en cada iteración, así como para ver el número de iteraciones producidas en cada prueba.

6.1.1. En el artículo



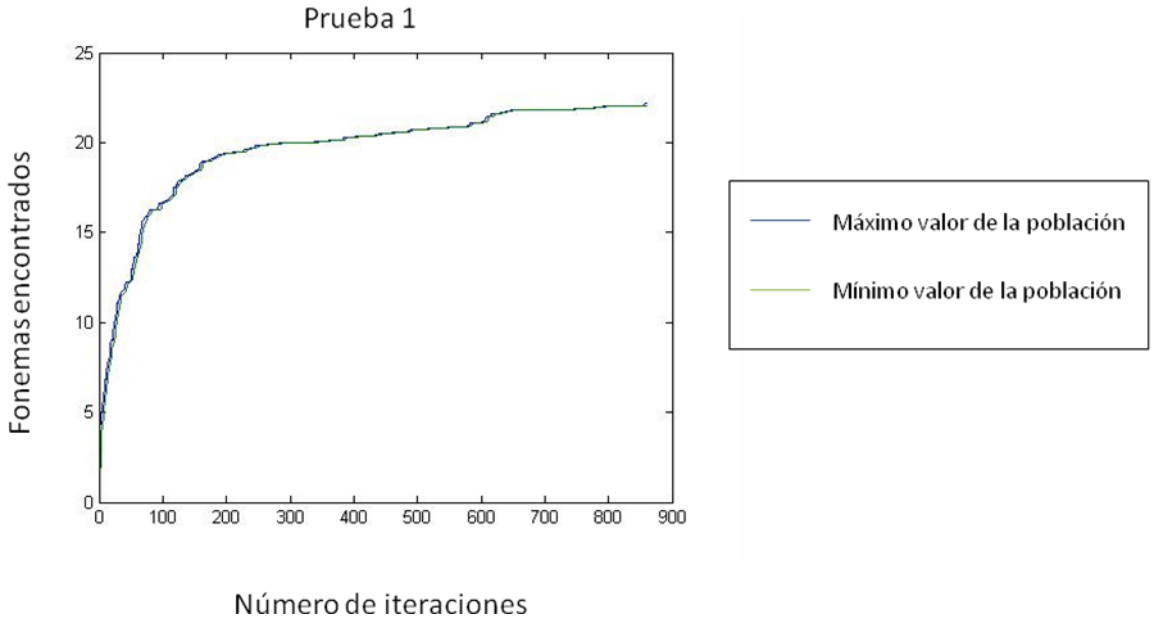


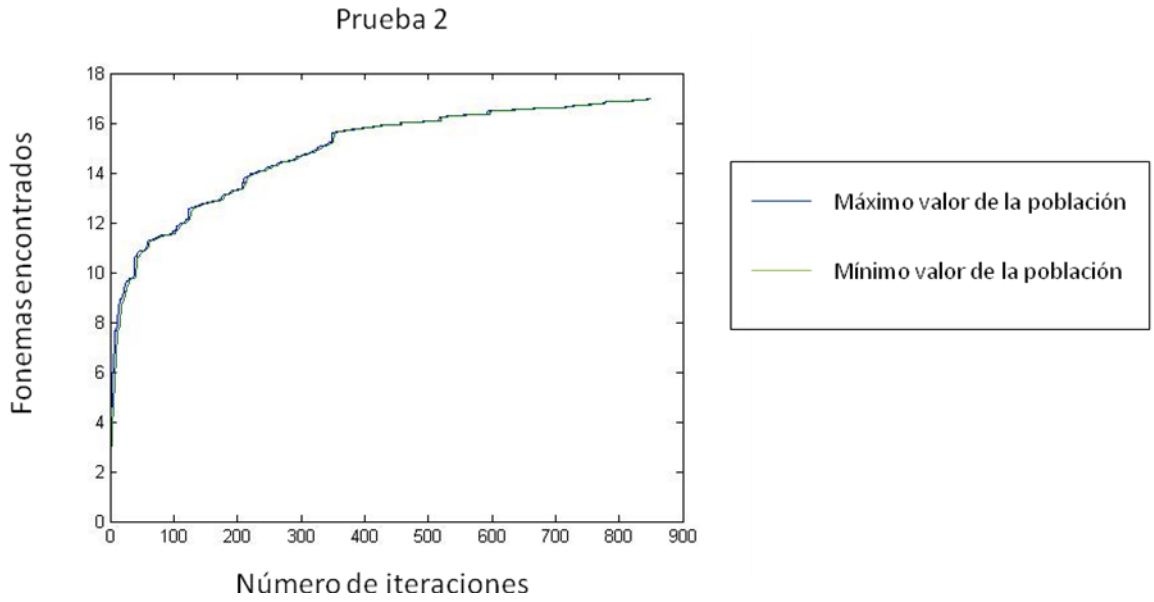
.

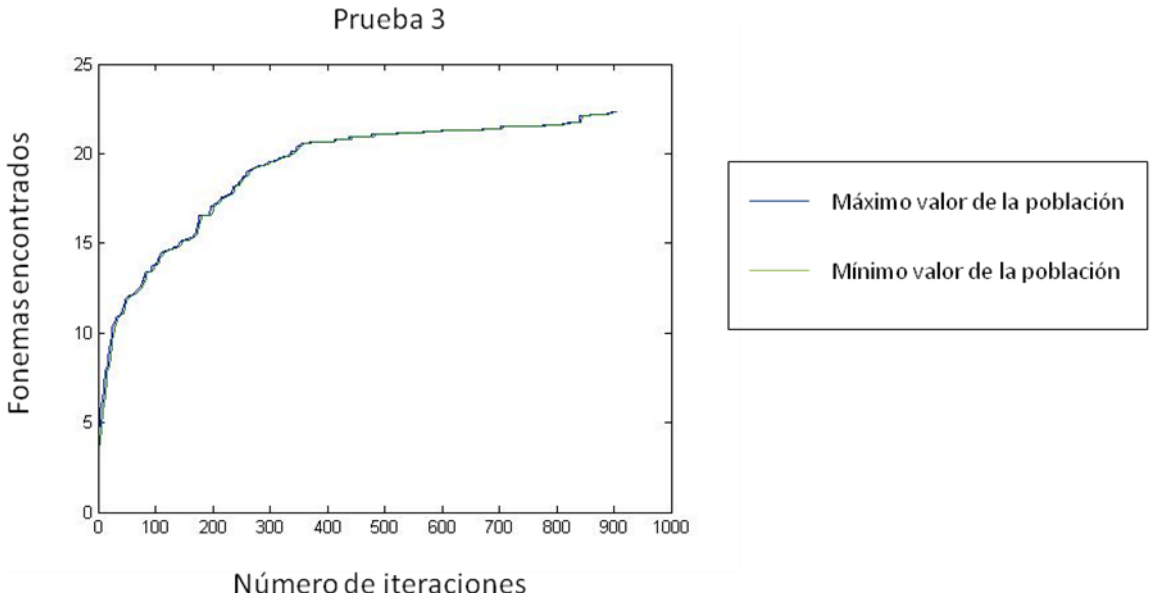


6.1.2. Extensión a Intervalos

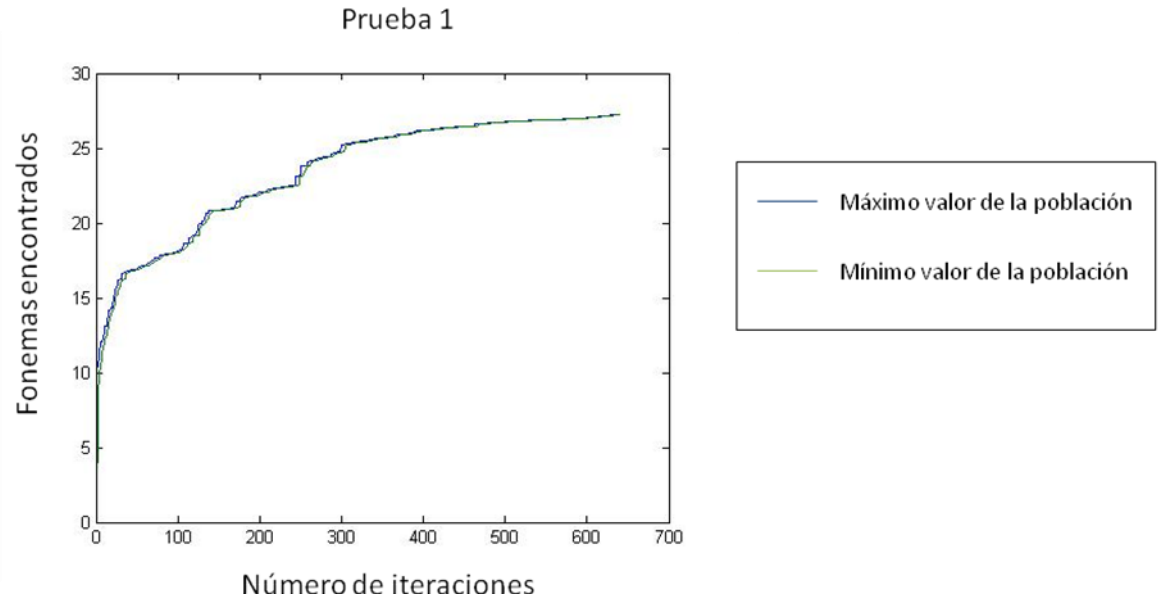
6.1.2.1. Intervalos en Datos

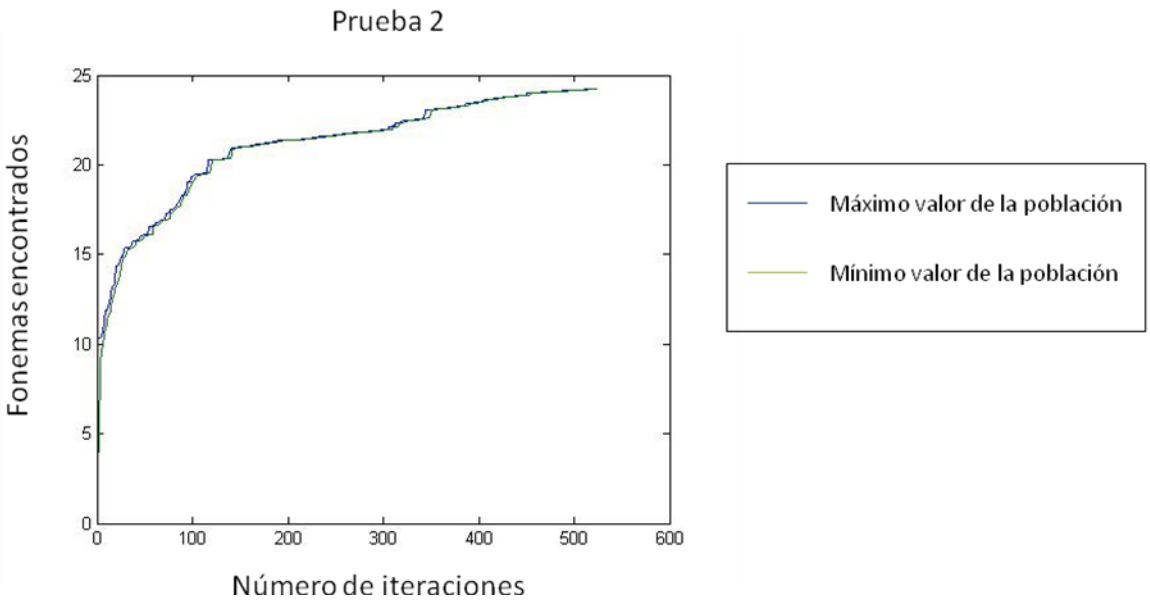


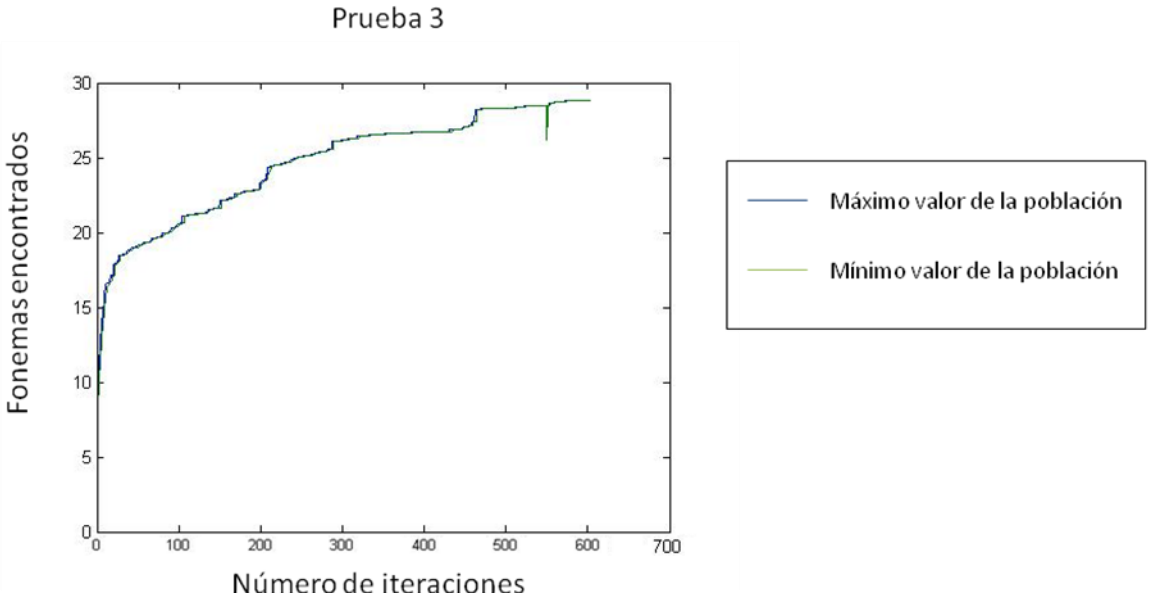




6.1.2.2. Intervalos en Funciones







6.2. Entrenamiento con 200 individuos por población

Esta prueba consiste en realizar un entrenamiento para cada método con 200 individuos por población en el que se aprecie la evolución del reconocimiento de los fonemas y las iteraciones producidas.

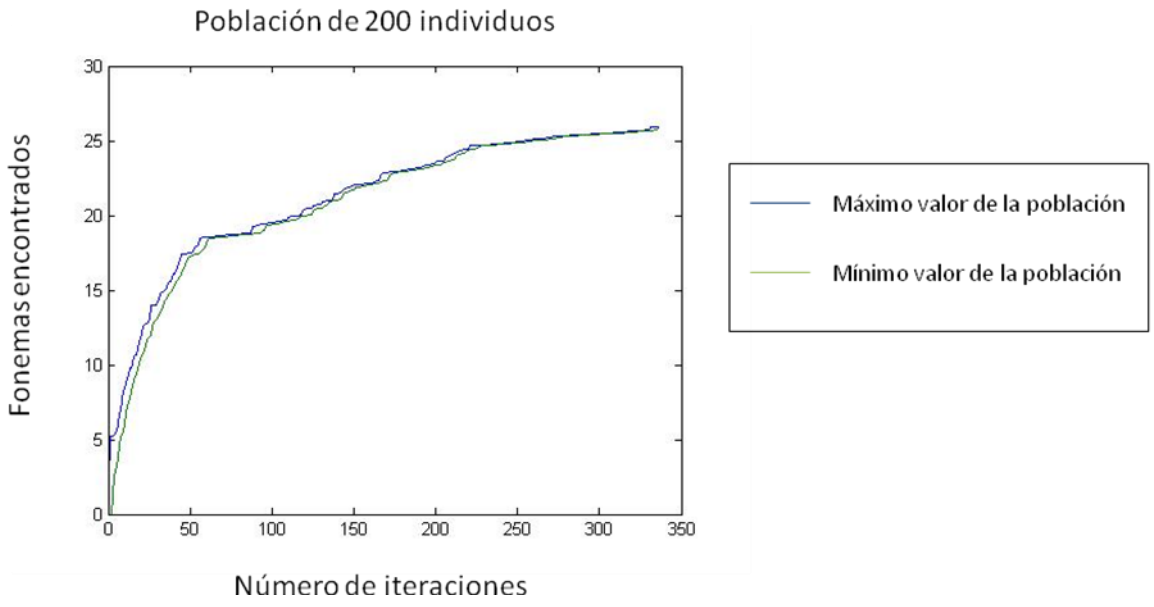
Se quiere comprobar si, a pesar de existir pocos individuos por población en los entrenamientos anteriores, los resultados son tan válidos como deberían ser con una población con un número de individuos más *adecuado*.

El número de individuos *adecuado* es igual o mayor al doble de genes que tiene cada individuo (si un individuo tiene 10 genes la población estaría correcta con 20 individuos).

Nota

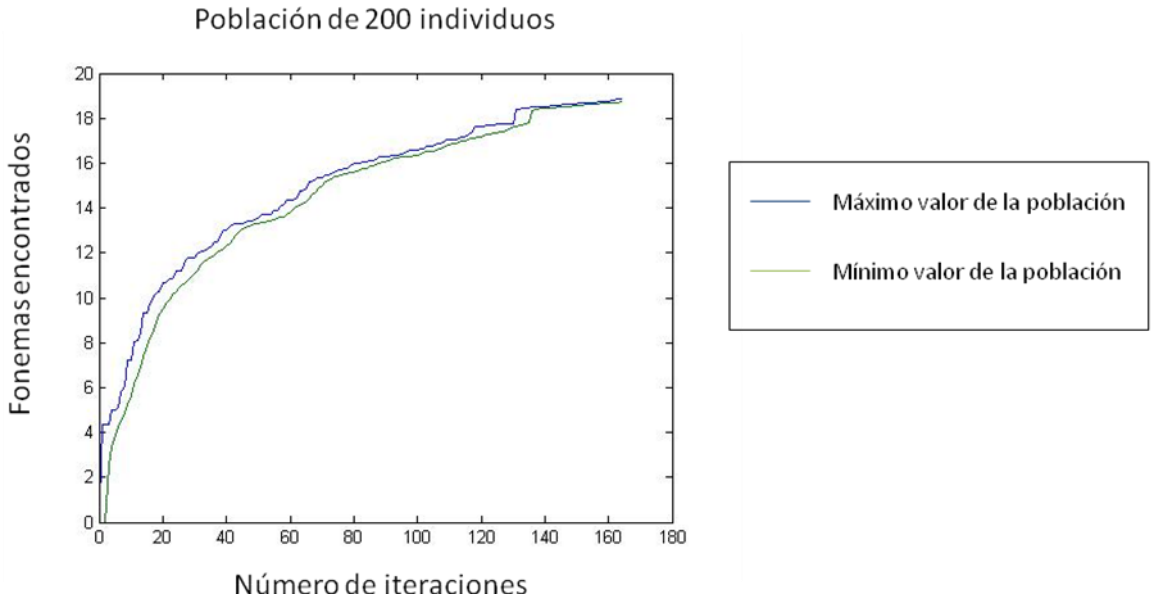
No se ha realizado esta segunda prueba con más individuos porque las máquinas con las que se ha trabajado no disponían de recursos suficientes.

6.2.1. En el artículo

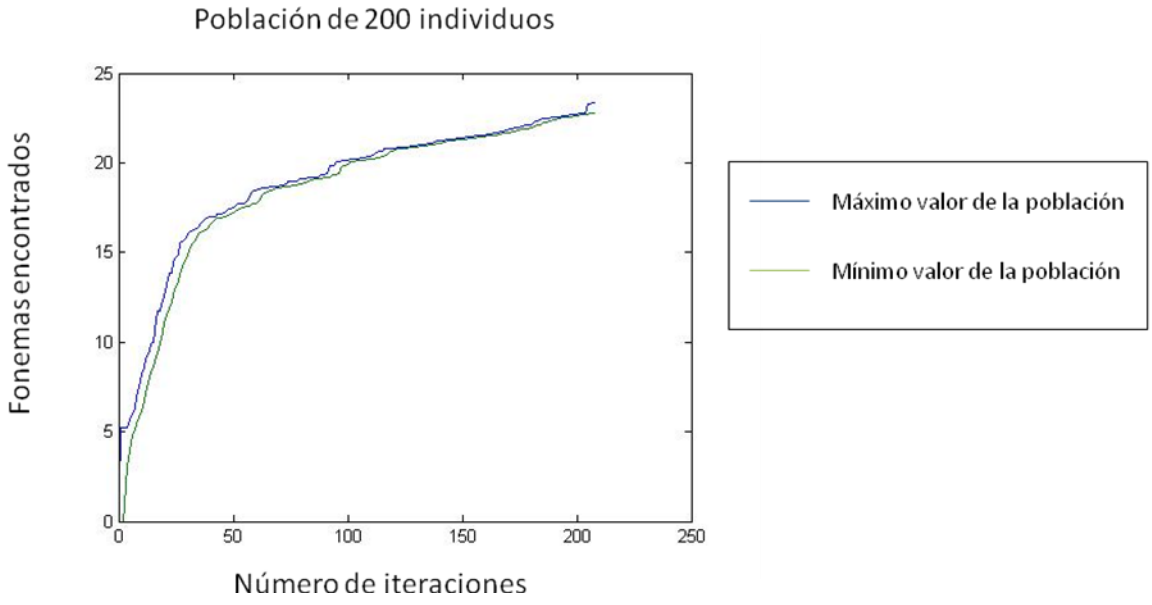


6.2.2. En extensión a Intervalos

6.2.2.1. Intervalos en Datos



6.2.2.2. Intervalos en Funciones



6.3. Reconocimiento en la muestra con los fonemas de las pruebas

Mediante la mejor definición de los fonemas que se ha encontrado en cada una de las pruebas, se realiza un reconocimiento de esos fonemas sobre la muestra en la que se ha entrenado. Además, se anotan los fonemas encontrados en cada bloque de la muestra con su porcentaje de similitud.

6.3.1. En el artículo

6.3.1.1. Prueba con 40 individuos 1

													Fonemas de la muestra																										
1	2	3	4	5	6	7	8	6	9	10	11	12	13	14	15	16	17	18	19	3	13	3	20	21	2	22	20	21	23	24	21	25	26	8	24
h	sh	ix	hv	eh	dcl	jh	ih	dcl	d	ah	kcl	k	s	ux	q	en	gcl	g	r	ix	s	ix	w	ao	sh	epi	w	ao	dx	axr	ao	l	y	ih	axr
													Fonemas encontrados (porcentaje)																										
1	2	3	4	10	6	7	8	6	9	10	11	12	13	3	19	16	17	18	19	3	13	3	8	21	2	22	25	21	23	24	21	25	26	8	24
100	100	100	100	29	100	100	57	100	94	64	100	100	100	63	100	100	100	100	100	70	100	100	100	83	96	100	100	96	100	90	97	100	100	100	92

6.3.1.2. Prueba con 40 individuos 2

													Fonemas de la muestra																										
1	2	3	4	5	6	7	8	6	9	10	11	12	13	14	15	16	17	18	19	3	13	3	20	21	2	22	20	21	23	24	21	25	26	8	24
h	sh	ix	hv	eh	dcl	jh	ih	dcl	d	ah	kcl	k	s	ux	q	en	gcl	g	r	ix	s	ix	w	ao	sh	epi	w	ao	dx	axr	ao	l	y	ih	axr
													Fonemas encontrados (porcentaje)																										
1	2	3	4	3	6	7	22	6	9	21	11	12	13	3	15	16	6	9	19	3	13	3	20	21	2	22	20	21	23	24	21	25	19	3	24
100	92	88	100	15	100	88	26	100	87	32	100	100	98	49	100	100	100	77	80	62	97	98	100	75	88	100	100	93	100	88	90	100	86	97	89

6.3.1.3. Prueba con 40 individuos 3

													Fonemas de la muestra																			
1	2	3	4	5	6	7	8	6	9	10	11	12	13	14	15	16	17	18	19	3	13	3	20	21	2	22	20	21	23	24	21	25	26	8	24
h	sh	ix	hv	eh	dcl	jh	ih	dcl	d	ah	kcl	k	s	ux	q	en	gcl	g	r	ix	s	ix	w	ao	sh	epi	w	ao	dx	axr	ao	l	y	ih	axr
													Fonemas encontrados (porcentaje)																						
1	7	3	4	3	6	7	14	6	9	14	11	12	13	14	15	16	17	18	19	3	13	3	19	21	11	22	14	21	23	24	21	25	19	3	24
100	100	100	100	21	100	100	49	100	100	25	100	100	100	74	100	100	100	100	100	69	100	100	100	84	98	100	100	99	100	100	100	100	99	99	100

6.3.1.4. Prueba con 200 individuos

													Fonemas de la muestra																			
1	2	3	4	5	6	7	8	6	9	10	11	12	13	14	15	16	17	18	19	3	13	3	20	21	2	22	20	21	23	24	21	25	26	8	24
h	sh	ix	hv	eh	dcl	jh	ih	dcl	d	ah	kcl	k	s	ux	q	en	gcl	g	r	ix	s	ix	w	ao	sh	epi	w	ao	dx	axr	ao	l	y	ih	axr
													Fonemas encontrados (porcentaje)																						
1	2	3	4	6	6	7	8	6	9	21	11	12	11	3	15	16	17	18	19	3	2	3	19	21	2	22	15	21	23	24	21	25	26	3	24
100	100	90	100	12	96	96	35	100	50	27	100	100	94	35	99	100	100	100	100	57	96	99	97	70	94	100	75	95	100	100	97	90	100	83	99

6.3.2. Extensión a Intervalos

6.3.2.1. Intervalos en Datos

6.3.2.1.1. Prueba con 40 individuos 1

													Fonemas de la muestra																			
1	2	3	4	5	6	7	8	6	9	10	11	12	13	14	15	16	17	18	19	3	13	3	20	21	2	22	20	21	23	24	21	25	26	8	24
h	sh	ix	hv	eh	dcl	jh	ih	dcl	d	ah	kcl	k	s	ux	q	en	gcl	g	r	ix	s	ix	w	ao	sh	epi	w	ao	dx	axr	ao	l	y	ih	axr
													Fonemas encontrados (porcentaje)																						
2	2	14	4	14	6	7	14	6	13	22	11	12	13	14	15	16	17	18	19	3	13	3	20	14	2	22	20	24	23	24	22	20	26	20	19
100	99	61	75	58	100	100	100	96	100	38	100	100	100	84	100	100	100	100	99	20	100	100	100	51	100	76	42	39	100	100	37	34	100	36	84

6.3.2.1.2. Prueba con 40 individuos 2

Para comprobar la evaluación obtenida de la población que se observa en la gráfica correspondiente [6.1.2. segunda gráfica], es necesario mostrar los fonemas encontrados más allá de la primera posición. En este caso se muestran hasta la tercera posición pero el fonema puede encontrarse en las siguientes con un alto porcentaje de similitud también.

													Fonemas de la muestra																									
1	2	3	4	5	6	7	8	6	9	10	11	12	13	14	15	16	17	18	19	3	13	3	20	21	2	22	20	21	23	24	21	25	26	8	24
h	sh	ix	hv	eh	dcl	jh	ih	dcl	d	ah	kcl	k	s	ux	q	en	gcl	g	r	ix	s	ix	w	ao	sh	epi	w	ao	dx	axr	ao	l	y	ih	axr
													Fonemas encontrados (porcentaje)																								
1	1	1	1	22	1	1	1	1	1	22	1	1	1	22	3	3	1	1	1	1	22	1	1	6	1	1	1	1	1	1	6	1	19	1	1
100	96	84	100	81	100	100	90	100	100	86	100	100	99	100	96	97	100	100	100	90	100	100	75	85	99	90	82	75	100	100	83	96	84	81	91
													Encontrados en 2º posición																								
13	7	9	4	6	2	22	22	6	6	12	13	12	22	12	15	1	2	2	19	6	1	3	3	22	22	6	21	9	6	21	22	25	15	15	3
100	92	63	51	70	100	91	57	100	100	73	100	100	99	81	95	96	100	100	77	46	99	97	71	84	98	78	41	57	100	44	81	41	83	51	86
													Encontrados en 3º posición																								
22	22	3	3	12	13	13	9	13	13	6	22	22	11	9	20	4	6	22	15	22	6	9	22	15	6	22	25	6	12	9	12	15	22	9	24
100	90	60	46	51	100	90	52	93	93	69	99	97	97	60	91	92	96	100	74	45	93	69	67	61	84	78	24	50	95	27	77	40	78	46	75

6.3.2.1.3. Prueba con 40 individuos 3

Al igual que en la prueba anterior, podemos encontrar que un fonema válido se encuentra más allá de la tercera posición.

													Fonemas de la muestra																						
1	2	3	4	5	6	7	8	6	9	10	11	12	13	14	15	16	17	18	19	3	13	3	20	21	2	22	20	21	23	24	21	25	26	8	24
h	sh	ix	hv	eh	dcl	jh	ih	dcl	d	ah	kcl	k	s	ux	q	en	gcl	g	r	ix	s	ix	w	ao	sh	epi	w	ao	dx	axr	ao	l	y	ih	axr
													Fonemas encontrados (porcentaje)																						
1	7	1	1	22	1	1	1	22	3	1	1	1	1	6	19	3	18	1	14	1	1	3	1	22	1	1	1	6	16	1	22	1	6	1	14
100	99	79	92	99	100	100	98	100	100	100	100	100	100	100	98	100	100	100	100	99	100	98	100	100	100	76	78	100	100	100	99	88	100	79	99
													Encontrados en 2º posición																						
6	1	3	3	6	6	6	6	1	16	6	6	18	6	22	3	19	22	11	19	22	6	1	6	6	6	11	17	22	1	24	6	3	22	6	24
100	99	73	53	92	100	100	73	98	98	100	100	99	100	100	97	100	99	100	98	73	100	98	100	98	100	72	50	100	93	44	98	47	100	66	96
													Encontrados en 3º posición																						
11	6	6	9	18	11	7	22	6	1	22	11	22	22	1	25	6	11	12	1	6	22	14	22	1	22	6	23	1	11	6	1	14	19	22	22
100	94	69	47	80	99	99	73	97	94	100	100	98	100	99	97	94	99	100	93	73	100	95	100	94	100	70	30	95	93	33	94	47	99	66	89

6.3.2.1.4. Prueba con 200 individuos

													Fonemas de la muestra																						
1	2	3	4	5	6	7	8	6	9	10	11	12	13	14	15	16	17	18	19	3	13	3	20	21	2	22	20	21	23	24	21	25	26	8	24
h	sh	ix	hv	eh	dcl	jh	ih	dcl	d	ah	kcl	k	s	ux	q	en	gcl	g	r	ix	s	ix	w	ao	sh	epi	w	ao	dx	axr	ao	l	y	ih	axr
													Fonemas encontrados (porcentaje)																						
2	2	3	4	22	6	7	16	6	2	22	11	12	13	6	15	16	17	18	19	6	13	3	20	6	2	22	20	24	23	24	6	20	26	20	17
17	93	20	75	34	97	90	21	100	58	48	100	75	95	52	73	99	85	79	78	20	99	50	76	41	85	73	26	34	100	91	29	32	92	24	65
													Encontrados en 2º posición																						
13	6	4	6	6	18	11	6	16	6	18	7	2	11	18	6	6	18	6	20	13	2	20	15	22	6	6	24	25	24	25	18	15	15	26	9
14	59	20	31	34	83	79	20	58	50	43	95	68	66	45	72	97	83	78	71	17	94	41	46	37	59	51	16	18	33	37	23	13	63	7	54
													Encontrados en 3º posición																						
7	7	20	9	18	9	6	20	18	9	6	6	13	6	22	18	9	9	17	26	18	22	19	3	18	11	2	25	8	25	20	24	19	19	25	18
11	58	19	5	34	83	72	19	47	50	41	62	66	63	40	69	78	75	68	48	13	80	29	13	24	59	45	11	0	21	9	13	13	58	4	45

6.3.2.2. Intervalos en Funciones

6.3.2.2.1. Prueba con 40 individuos 1

													Fonemas de la muestra																										
1	2	3	4	5	6	7	8	6	9	10	11	12	13	14	15	16	17	18	19	3	13	3	20	21	2	22	20	21	23	24	21	25	26	8	24
h	sh	ix	hv	eh	dcl	jh	ih	dcl	d	ah	kcl	k	s	ux	q	en	gcl	g	r	ix	s	ix	w	ao	sh	epi	w	ao	dx	axr	ao	l	y	ih	axr
													Fonemas encontrados (porcentaje)																										
11	2	4	4	7	17	17	22	22	9	7	17	22	11	21	17	11	17	18	17	11	11	17	21	21	2	22	25	21	23	17	21	25	11	21	17
9	82	23	94	12	96	62	8	34	27	21	70	73	86	38	43	80	93	29	22	21	91	57	51	67	94	88	55	88	33	66	85	84	23	23	34
													Encontrados en 2º posición																										
7	7	7	17	2	22	22	21	17	18	11	22	12	7	15	25	17	22	17	22	4	2	4	4	11	7	17	22	25	9	21	11	22	21	17	11
3	80	20	44	9	73	58	8	29	11	21	60	66	71	25	36	63	50	15	18	20	85	54	9	14	81	51	54	45	27	40	55	53	16	22	33
													Encontrados en 3º posición																										
2	11	11	7	11	4	7	4	18	22	21	11	17	2	3	4	4	4	22	21	17	7	22	11	2	11	4	23	11	4	4	3	9	22	22	7
3	71	16	19	9	55	52	5	21	8	15	54	48	62	21	30	58	36	10	16	15	75	51	6	13	80	23	33	40	17	35	27	23	12	20	29

6.3.2.2.2. Prueba con 40 individuos 2

													Fonemas de la muestra																										
1	2	3	4	5	6	7	8	6	9	10	11	12	13	14	15	16	17	18	19	3	13	3	20	21	2	22	20	21	23	24	21	25	26	8	24
h	sh	ix	hv	eh	dcl	jh	ih	dcl	d	ah	kcl	k	s	ux	q	en	gcl	g	r	ix	s	ix	w	ao	sh	epi	w	ao	dx	axr	ao	l	y	ih	axr
													Fonemas encontrados (porcentaje)																										
6	22	22	4	6	17	22	20	17	9	20	12	12	22	20	17	18	17	17	20	12	22	17	20	20	12	22	22	20	23	22	20	22	20	20	22
83	52	36	74	2	57	49	14	71	71	18	55	100	24	34	58	46	81	89	36	18	40	53	97	37	36	86	54	96	92	70	89	53	32	49	52
													Encontrados en 2º posición																										
1	18	4	12	3	12	18	17	22	22	3	22	22	12	3	12	22	22	12	12	22	12	22	17	10	22	18	20	22	22	18	24	17	4	4	18
57	44	32	43	1	46	37	13	42	52	4	43	83	22	13	52	45	73	87	33	17	38	39	15	12	36	61	50	39	87	62	15	53	4	27	52
													Encontrados en 3º posición																										
2	12	18	22	24	22	12	12	18	18	6	18	18	18	19	22	17	12	18	17	20	18	18	22	21	18	12	23	18	18	23	25	20	22	22	20
57	35	24	41	1	46	36	11	29	44	3	34	71	17	9	47	38	71	76	26	15	34	37	8	11	22	40	48	34	75	56	13	46	3	25	51

6.3.2.2.3. Prueba con 40 individuos 3

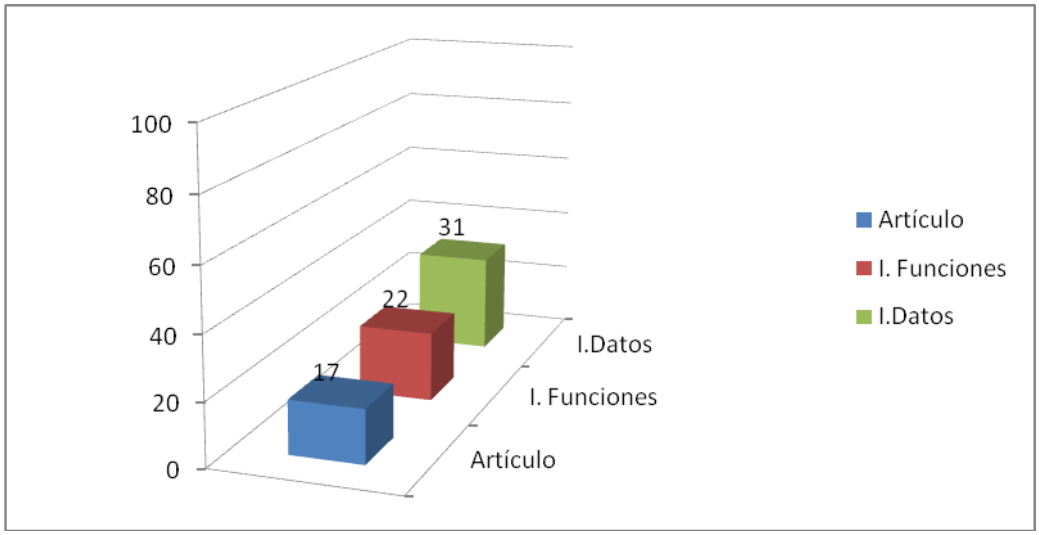
													Fonemas de la muestra																									
1	2	3	4	5	6	7	8	6	9	10	11	12	13	14	15	16	17	18	19	3	13	3	20	21	2	22	20	21	23	24	21	25	26	8	24
h	sh	ix	hv	eh	dcl	jh	ih	dcl	d	ah	kcl	k	s	ux	q	en	gcl	g	r	ix	s	ix	w	ao	sh	epi	w	ao	dx	axr	ao	l	y	ih	axr
													Fonemas encontrados (porcentaje)																									
6	22	22	4	6	17	22	20	17	9	20	12	12	22	20	17	20	17	17	20	4	22	17	20	20	22	22	22	20	23	20	20	22	20	20	20
86	56	39	80	3	57	52	18	68	71	19	59	99	34	35	56	51	82	87	68	26	46	57	98	39	44	89	57	94	92	85	91	58	32	68	57
													Encontrados en 2º posición																									
1	12	4	22	3	12	12	17	22	22	7	22	22	12	3	12	22	22	12	12	12	12	20	17	21	12	18	17	22	22	22	21	17	26	4	22
57	38	37	43	2	50	41	13	43	55	6	46	83	32	13	55	49	74	85	36	21	45	54	14	16	43	54	54	46	90	74	24	55	12	32	56
													Encontrados en 3º posición																									
2	4	23	12	24	4	18	12	12	17	3	18	18	20	19	22	4	12	22	17	20	4	4	12	10	4	12	23	4	17	23	25	20	22	22	4
57	26	15	33	2	50	27	11	25	44	6	25	62	12	9	49	47	71	74	27	21	20	44	13	15	20	42	50	22	81	55	20	46	9	30	56

6.3.2.2.4. Prueba con 200 individuos

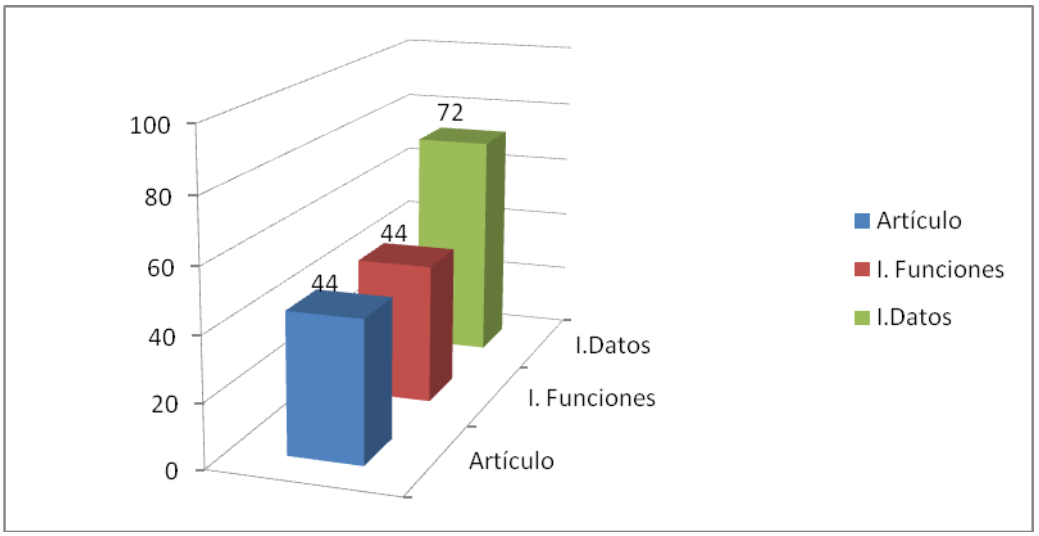
													Fonemas de la muestra																									
1	2	3	4	5	6	7	8	6	9	10	11	12	13	14	15	16	17	18	19	3	13	3	20	21	2	22	20	21	23	24	21	25	26	8	24
h	sh	ix	hv	eh	dcl	jh	ih	dcl	d	ah	kcl	k	s	ux	q	en	gcl	g	r	ix	s	ix	w	ao	sh	epi	w	ao	dx	axr	ao	l	y	ih	axr
													Fonemas encontrados (porcentaje)																									
1	2	3	2	2	12	2	2	2	9	2	12	12	2	3	25	2	18	2	3	3	2	3	25	21	2	12	21	21	9	3	21	25	25	3	3
100	98	87	67	10	87	96	21	77	55	19	95	99	91	19	60	92	83	42	70	50	97	97	81	67	94	63	65	88	56	77	84	91	32	60	81
													Encontrados en 2º posición																									
2	7	2	3	18	18	12	3	12	23	3	2	18	12	19	3	25	2	9	25	2	7	2	3	25	12	7	3	23	23	25	23	3	19	25	18
70	90	40	57	6	75	81	16	40	22	11	80	80	32	16	56	74	66	35	35	30	63	77	70	12	79	59	61	78	50	68	74	48	10	49	51
													Encontrados en 3º posición																									
6	12	25	18	7	7	7	23	7	2	18	7	7	18	23	2	18	12	23	21	25	12	25	21	2	7	2	25	25	2	21	25	21	2	21	2
61	81	21	23	5	66	79	12	39	17	10	61	59	21	12	53	69	63	2	23	25	62	75	58	12	78	44	54	73	14	63	41	45	9	30	50

6.4. Reconocimiento sobre la muestra de voz alterada

Se ha alterado el volumen de la muestra para probar el reconocimiento de voz sobre una grabación algo distinta. Para esta prueba se ha utilizado la mejor definición de los fonemas obtenida por el entrenamiento en cada método.



Número de fonemas localizados en las 3 primeras posiciones para una grabación con un tono 5 veces mayor que el usado en el entrenamiento.



Número de fonemas localizados en las 3 primeras posiciones para una grabación con un tono ligeramente menor que la muestra del entrenamiento.

7. Conclusiones

7.1. Del proceso de entrenamiento

En las gráficas del primer experimento [6.1] se observa que en todos los métodos la evolución del mejor caso crece de forma logarítmica. Primero aumenta muy rápidamente encontrando enseguida valores mejores a los existentes y produciendo la pronunciada curvatura en las gráficas y después frena paulatinamente hasta llegar a mantenerse en un estado durante un largo periodo de tiempo.

Esto indica que el entrenamiento es un proceso duradero, ya que llegados a un punto, la diferencia entre un buen fonema y un mal fonema se encuentra en unos pocos genes que lo definen. Debido a la cantidad de información que tiene el genoma de un individuo, conseguir que el azar mute el gen preciso puede llevar cientos de generaciones.

Además, como se puede notar si se comparan las gráficas de los entrenamientos con 40 individuos y los entrenamientos con 200 individuos, en los entrenamientos con 40 individuos la diferencia entre el valor máximo y mínimo en cada iteración es muy pequeña. Esto hace que en una misma población haya muy poca variabilidad y por tanto el crecimiento sea más lento.

En la gráfica de 200 individuos, para un mismo tiempo de tres horas y media se observa que se han realizado muchas menos iteraciones porque se trabaja con más datos, sin embargo, en menos iteraciones se ha conseguido una mejor evaluación gracias a la variabilidad de los individuos de la misma población.

Nota

En futuras versiones se intentará disponer de mejores máquinas para trabajar con una mayor población y conseguir una mayor variabilidad.

7.2. Del tiempo de entrenamiento

Se puede observar, al comparar las gráficas del ejemplo del artículo con las de los métodos intervalares, que sí existe una diferencia real en el tiempo empleado por un método y otro. En el ejemplo del artículo se ha realizado una media de 1700 iteraciones por prueba, mientras que en los métodos de intervalos han rondado las 600-800 iteraciones. Se deduce de esta información que los modelos de intervalos son prácticamente la mitad de rápidos que el modelo del artículo (se pueden comparar estos resultados porque todas las pruebas han funcionado durante el mismo tiempo, entre 3 horas y media y 4 horas).

La diferencia de tiempo entre los tres modelos explicados pierde importancia si se trata de aplicar el reconocimiento de voz en una circunstancia normal (conversación), ya que la tecnología y las máquinas actuales son capaces de trabajar con una gran cantidad de información en poco tiempo. Toda la información de una conversación (Ejemplo: un espectrograma) se puede someter a los procesos de reconocimiento explicados en este documento, y al instante, sin notarse diferencia (para la percepción del ser humano) entre el reconocimiento del tipo del artículo y el reconocimiento con intervalos, devolver los resultados.

Nota

Al hacer las pruebas, se ha llevado a cabo el proceso de reconocimiento en los tres método y no se ha notado diferencia alguna en el tiempo de ejecución (reconocimiento(), reconocimientoDatos() y reconocimientoFunciones()).

Si los métodos con intervalos mejorasen los resultados conocidos de otros reconocedores de voz, sería un buen sustituto para palabras o frases cortas ya que los seres humanos no nos percataríamos de la pérdida de velocidad.

7.3. Memoria utilizada en el entrenamiento

Del mismo modo que con el tiempo, se puede comprobar por medio de las gráficas del apartado [6.1.], que los métodos por intervalos requieren de mucha más memoria que el método del artículo. El hecho de que en intervalos se hayan producido casi la mitad de iteraciones que en el artículo, indica que se ha tardado más tiempo en evaluar una población. Este tiempo de más se debe a que, para evaluar cada población, se ha tenido que trabajar con muchos más datos.

Sin embargo, el consumo de memoria no tiene demasiada importancia si lo que se desea es trabajar con muestras de conversaciones normales (Ejemplo de conversación: *En internet está la página de la Universidad Pública de Navarra*), ya que las máquinas actuales están diseñadas para trabajar con bloques de datos de mayor tamaño que el necesario para reconocer una frase o palabra.

7.4. Resultados obtenidos

A lo largo del documento sólo se ha podido trabajar con una muestra de ejemplo (LDC93S1.wav; *She had her dark suit in greasy wash water all year*). Ha resultado un problema ya que nos ha obligado a centrarnos principalmente en pruebas de eficiencia de los métodos (tiempo y memoria). Como el reconocimiento es un proceso sencillo y breve, no se podían realizar las pruebas sobre él teniendo que hacerlas en el proceso de entrenamiento.

Para poder realizar comparaciones reales en base a la eficiencia, todos los métodos han estado funcionando durante un mismo período de tiempo. Las pruebas finales de reconocimiento se han visto influenciadas por este hecho ya que, al producirse en los métodos intervalares menos iteraciones que en el artículo, los fonemas no se encontraban tan bien ajustados. Se puede observar en las tablas del apartado [6.3.] que en los casos de intervalos se detectan menos fonemas correctos en las primeras posiciones (por ese motivo se han mostrado los 3 primeros fonemas en cada bloque).

También hay que decir que, a pesar de encontrar menos fonemas en las primeras posiciones, se detectan muchos más fonemas con altos valores de coincidencia (esto podría resultar útil si se desease, por ejemplo, escribir un mensaje, de forma que al decir

una palabra, apareciese una lista con todas las palabras similares y pudieses elegir una). Si no se tiene en cuenta la posición en la que se encuentra un fonema y nos fijamos solo en los porcentajes de coincidencia, se obtienen buenos resultados, sobre todo en el caso de intervalos en datos, a pesar de haberse sometido a menos ajustes en la definición de sus fonemas (iteraciones en el entrenamiento).

Si se añade al comentario anterior, que en la prueba con la muestra de voz modificada, el método en el que hay más fonemas en las primeras posiciones es intervalos en datos, se puede pensar que es el camino con mejor previsión de cara al futuro.

Sin embargo, la falta de pruebas con diferentes muestras de voz nos impide decantarnos con total certeza, ya que es posible, mediante un buen entrenamiento con varias grabaciones conocidas distintas, que con cualquiera de los otros dos métodos estudiados (artículo e intervalos en funciones) se obtengan mejores resultados.

8. Bibliografía

ARTICULOS.

Anónimo (2004). ***Reconocedor de patrones sonoros*.**

Elwakdy A. M., Elsehely B.E., Elthokhy C.M., Elhennawy D.A.*.* (2008). ***Speech recognition using a wavelet transform to establish fuzzy inference system through subtractive clustering and neural network (ANFIS).*** International journal of circuits, systems and signal processing.

Engin Avci, Zuhtu Hakan Akpolat. (2006). ***Speech recognition using a wavelet packet adaptive network based fuzzy inference system.*** Expert Systems with Applications 31 495–503, Department of Electronic and Computer Science, Firat University,Turkey.

Garmendia L., Ortiz F., ROJAS T., VARGAS A., VELASQUEZ R. ***Normas Y Conormas de la lógica difusa***.

González del Campo, R. ***Conjuntos difusos intervalo-valorados: estado del arte***. Jornadas Internacionales de Didáctica de las Matemáticas en Ingeniería

Halavati R, Bagheri Shouraki S., Harati Zadeh S. (2006). ***Recognition of human speech phonemes using a novel fuzzy approach.*** Artificial Intelligence Lab 308, Computer Engineering Department, Sharif University of Technology, Tehran, Iran.

Halavati R., Shouraki S B., Eshraghi M., Alemzadeh M., Ziaie P., (2004). ***A Novel Fuzzy Approach to Speech Recognition.*** *Computer Engineering Department,*Sharif University of Technology, Tehran, Iran. Proceedings of the Fourth International Conference on Hybrid Intelligent Systems (HIS'04).

Hernandez-Abrego G. and Mariño J.B.(1999). ***Fuzzy reasoning in confidence evaluation of speech recognition.*** TALP Research Center Dept Teoria del Senyal i Comunicacions, Universitat Politécnica de Cataluña.

Kasabov N.K., Kozma R., Watts M.J. (1998). ***Phoneme-based speech recognition via fuzzy neural networks modeling and learning.*** Department of Information Science, University of Otago, New Zealand(1997).al. / information Sciences 110, 61-79.

Martinez, L. (1999). ***Un nuevo modelo de representación de información lingüística basado en 2-tuplas para la agregación de preferencias lingüísticas.*** Phd Thesis, Universidad de Granada.

Mills P, Bowles J**.** (1996). ***Fuzzy Logic –Enhanced Symmetric Dynamic Programming for Speech Recognition.*** University of South Carolina, Electrical and Computer Engineering Columbia

Momtazi S., Sameti H., Bahrani M., Hafezi N.(2007). ***A pos-based fuzzy word clustering algorithm for continuous speech recognition systems.*** Speech Processing Lab, Computer Enginneering Department, Sharif University of Technology, Teheran, Iran.

Oppizzi O., Fournier D., Gilles P., Meloni Ceri H. (1996). ***A fuzzy acoustic-phonetic decoder for speech recognition*****.** Laboratoire d'informatique, Avignon France.

Orduna R. (2010). ***Funciones de solapamiento y su aplicación a segmentación de imágenes.*** *Phd Thesis,* Universidad Pública Navarra.

Pagola, M. (2008). ***Representation of Uncertainity by Interval Valued Fuzzy Sets. Application to Image Thresholding***. Phd Thesis, Universidad Pública Navarra.

Rodríguez W.(2000) . ***Fuzzy Logic Based Voice Procesing.*** Departamento de Computación Facultad de Ingeniería Universidad de Los Andes Mérida, Venezuela.

Tzanetakis G., Cook P.; (2002). ***Musical Genre Classification of Audio Signals.*** IEEE Transaction on Speech and Audio Processing, vol. 10, NO. 5,

Zhao Peng-Yung Woo T. (1996). ***Fuzzy Speech Recognition.*** Electrical Engineering Department Northern Illinois University.

Zhang X, Zbigniew W. Ras.*(2007).* ***Analysis of Sound Features for Music Timbre Recognition.*** Department of Computer Science University of North Carolina at Charlotte.

MANUALES

Manual de la toolbox de MATLAB Signal Processing Toolbox

PÁGINAS WEB CONSULTADAS

http://personal.telefonica.terra.es/web/sixsancal/documentacion/Pdf por partes/3-2_El espectrograma.pdf

http:/giara.unavarra.es/

http://es.wikipedia.org/wiki/L%C3%B3gica_difus http://ans

http://es.wikipedia.org/wiki/Conjunto_difusowermath.com/logica_difusa_conjuntos_nebulosos.htmamels.htmlojects2.
http://delta.cs.cinvestav.mx/~gmorales/ldifl/node9.htmlphp

http://www.bioingenieria.edu.ar/grupos/cibernetica/milone/download/hmmdiet.pdf

http://www.iac.es/sieinvens/SINFIN/Sie_Courses_PDFs/NNets/confiac.pdf

http://www.uoc.edu/in3/emath/docs/Componentes_principales.pdf

Ingeniería Técnica en Informática de Gestión

USOS DE LA LÓGICA DIFUSA E INTERVALOS EN EL RECONOCIMIENTO DEL HABLA

Juan Cerrón González

Funciones generales

```
function [black blue purple red yellow green cyan white maximo1
maximo2]=aColores(S)
%Función que crea todas las matrices para cada uno de los colores a
partir
%de la matriz principal.
black=generalizada(S,0,0,0.05,0.35);
blue=generalizada(S,0,0.05,0.22,0.5);
purple=generalizada(S,0,0.23,0.35,0.63);
red=generalizada(S,0.05,0.36,0.50,0.80);
yellow=generalizada(S,0.23,0.51,0.63,0.93);
green=generalizada(S,0.36,0.63,0.80,0.99);
cyan=generalizada(S,0.51,0.81,0.93,0.99);
white=generalizada(S,0.63,0.94,0.99,0.99);

%Paso todas estas matrices por la función cogeMax para que me cree dos
%matrices. Una con los colores máximos (representados por números) y
otra
%con los segundos máximos.
[ maximo1 maximo2 ]=
cogeMax(black,blue,purple,red,yellow,green,cyan,white);
%El orden en el que se pasen los colores influirá a la hora de elegir
que
%número corresponde a cada color.       cogeMax(1,2,3,4,5,6,7,8)
```

```
function longitud=aLongitud(phonem,long)
%esta funcion devuelve en 'longitud' el fonema "phonem" en el que
hemos
%añadido una fila más (al final) que hace referencia a la longitud del
%fonema.
longitud=phonem;

%Aplicamos la ecuación generalizada de la recta con los parámetros que
%corresponden a cada longitud.
vsl=generalizada(long,2,2,3,5); %very short length
sl=generalizada(long,2,2,6,10); %short length
ml=generalizada(long,2,6,12,17); %medium length
ll=generalizada(long,8,12,20,30); %long length
vll=generalizada(long,12,18,69,99); %very short length

vector = [ vsl sl ml ll vll ]; %Vector con los diferentes valores de
longitud del fonema.

vector2=find(vector==max(vector)); %Como me puede devolver varios
índices,
        %los guardo en un vector para coger más adelante el primero.

%Guardo en la matriz 'longitud' el fonema añadiendo una fila más con
un
%número correspondiente a la longitud.
longitud(size(longitud,1)+1,1)=vector2(1);
```

```
function [maxima1
maxima2]=cogeMax(black,blue,purple,red,yellow,green,cyan,white)
%Función que coge los dos colores en los que se ha obtenido un mayor
valor
%de pertenencia.

%La lógica que se sigue es que cada posición se va a corresponder
siempre a
%una matriz en cuestión. Cada matriz se reconocerá por el índice en el
que
%se encuentra dentro de vector. La posición 1 será para negro, la 2
para
%azul...

for i=1:size(black,1) %Filas
    for j=1:size(black,2) %Columnas

        vector = [black(i,j) blue(i,j) purple(i,j) red(i,j)
yellow(i,j) green(i,j) cyan(i,j) white(i,j)]; %A es 1(negro); B es
2(morado); C es 3(azul);

        vector2=find(vector==max(vector)); %Como me puede devolver
varios índices,
        %los guardo en un vector para coger más adelante el primero.

        maxima1(i,j)=vector2(1); %Guardo el índice de donde se
encontraba el máximo

        vector(vector2(1))=-1; %Pongo al mínimo el primer valor máximo
para poder coger el siguiente máximo como el 2°.

        %Si el siguiente máximo que cogemos es 0 quiere decir que no
hay
        %valor válido, así que volvemos a guardar el del máximo
primero. Si
        %por el contrario es distinto de 0, lo guardamos como
siguiente
        %máximo.
        if (max(vector)~=0)
           vector2=(find(vector==max(vector))); %Vuelvo a buscar la
posición del máximo(esta vex el 2° ya que
        end

        maxima2(i,j)=vector2(1); %Guardo el índice de donde se
encontraba el máximo
    end
end
```

```
function y=coloresTipoEntrenamiento(tipo,S,matrizDefine)
%Función que para un tipo dado de color (1==negro, 2==azul...) y un
valor
%S te dice el valor que le corresponde.

%La matrizDefine tiene el valor correspondiente a lo que hay que
pasarle a
%la función generalizada.

%  negro=>(1) [0,0,0.05,0.35]
%  blue =>(2) [0,0.05,0.22,0.5]
%  ...

%Como coincide que el tipo(1=negro,2=blue,3,4,5...) es = a la fila en
la que se encuentran
%los valores en la matrix matrizDefine, podemos poner una sola línea
en

y=generalizada(S,matrizDefine(tipo,1),matrizDefine(tipo,2),matrizDefin
e(tipo,3),matrizDefine(tipo,4));



function y=longitudTipoEntrenamiento(tipo,long,matrizDefine)
%Como en matrizDefine se que tengo 8 filas para las columnas y después
las
%5 filas de  las longitudes, puedo poner aqui una sola línea con el
índice
%de matrizDefine=tipo+8

y=generalizada(long,matrizDefine(tipo+8,1),matrizDefine(tipo+8,2),matr
izDefine(tipo+8,3),matrizDefine(tipo+8,4));


function m = freq2mel (f)
% compute mel value from frequency f
m = 2595 * log10(1 + f./700);


function sol=extraePos(maxGenoma)
%Genera una posición al azar entre 1 y maxGenoma

%rand()genera el número del que queremos obtener la posición.
%maxGenoma es el máximo valor que puede devolver esta función al hacer
la
%regla de 3 con el numero.
numero=rand();
if numero==0
    numero=rand(); %Como la probabilidad de obtener dos veces seguidas
0 es muy baja así estaría resuelto.
    if numero==0
        numero=numero+0.1; %por si acaso me vuelve a salir, sumo 0.1
    end
end

sol=ceil(numero*maxGenoma); %ceil redondea hacia el entero más alto.
%si numero es 0.8 y los posibles valores llegan hasta el 40(maxGenoma)
%sol sería el 32, luego se correspondería con algún fonema.
```

```
function [Snew fnew tnew] = grab2mel(XY)
%Transforma las frecuencias del espectrograma que se obtiene de la
%grabación LDC93S1.wav según la escala de Mel para aproximarlas a las
%frecuencias mejor percibidas por el ser humano.

%Grabación que queremos tratar.
XX='LDC93S1.wav';

%Guardamos en una variable el nombre de la grabación y su frecuencia.
[grab fs]=wavread(XX);

%Hacemos el spectrograma y guardamos las frecuencias FFT y los ejes X
e Y
%(tiempo y frecuencias)
[S f t]=specgram(grab,XY,fs);
Snew=S*0; % Defino la matriz que voy a devolver inicializándola a 0.

%Paso la escala de las frecuencias "f" a la escala en mel.
for k=1:size(f,1)
    fnew(k,1)=abs(freq2mel(f(k))); %Cojo el absoluto para no tener en
cuenta la parte compleja (i)
end

%Creo la tabla de frecuencias con igual proporción de intervales entre
%0-2840
inter=floor(max(fnew)/size(fnew,1));
for k=0:size(fnew,1)-1
    fnew2(k+1,1)=k*inter; %Cojo el absoluto para no tener en cuenta la
parte compleja (i)
end

for i=1:size(fnew2,1) %voy a copiar cada fila de la matriz S en el
lugar que le correspondería según la escala de Mel
    j=1; %Inicializo la variable j que usamos para cada valor de i.
    while fnew2(i)>floor(fnew(j))
        j=j+1;
    end
    %Si coincide la frecuencia de mel con la frecuencia del intervalo
(cada
    %44Mel) sustituyo la fila entera.
    if fnew2(i)==floor(fnew(j))
        Snew(i,:)=S(j,:);
    else
    %Si el intervalo se encuentra entre dos frecuencias de mel cojo
    %posición a posición la mayor intensidad de cada tiempo comparando
        %ambas frecuencias.
        for h=1:size(Snew,2)
            Snew(i,h)=max(S(j-1,h),S(j,h));
        end
    end
end

%Paso las frecuencias nuevamente a frecuencias de 0 a 8000. Esta vez
los
%intervalos no serán idénticos.
for k=1:size(f,1)
    fnew(k,1)=abs(mel2freq(fnew2(k))); %Cojo el absoluto para no tener
en cuenta la parte compleja (i)
end
tnew=t;
```

```
fnew=flipud(fnew);
figure;imagesc(tnew,fnew,20*log10((abs(Snew))));
Snew
```

```
function b = mel2freq (m)
% compute frequency from mel value
b = 700*((10.^(m ./2595)) -1);
```

```
function y=generalizada(S,Start,First,Last,End)
%Es la función que se encarga de realizar la ecuación normal de la
recta.

y=S;

%Primero inicializamos los puntos que conocemos
(Start,First,Last,End). El
%máximo será 1 en la 'y' y 0.99 en 'x'
y(y<=Start)=0; %Inicial
y(y==First)=1; %Primer máximo
y(y==Last)=1; %Último máximo
y(y>=End)=0; %Final


%La ecuación de la recta entre dos puntos queda así para Start y
%First(cambiandolos obtenemos todos los casos).
%
%        (X   - Start)
% y = -------------------
%       (First - Start)
%
%Donde X es cada punto de la matriz e y el valor de la nueva matriz.


%Con Start y First
if Start~=First
    y(y>Start & y<First)=(y(y>Start & y<First)-Start) / (First-Start);
end

%Con First y Last
if First~=Last
    %Si el máximo entendemos que es 1, tanto en el primero como el
último
    %será 1, y por tanto cualquier punto entre ambos máximos será
también
    %1.
    y(y>First & y<Last)=1;
end

%Con Last y End
if Last~=End
    y(y>Last & y<End)=(End - y(y>Last & y<End)) / (End-Last);
end
```

Específicas para crear la primera inicialización de los fonemas

```
function y=speechreduction(S,max10)
%devuelve la matriz reducida.

%Guardo el intervalo con el que voy a generar las 25 bandas.
inter=floor(size(S,1)/25);

%'j' va a comenzar en 1 + "intervalo", ya que queremos coger la
primera vez
%desde 1 hasta j para hacer la media.
j=inter;
while j<=size(S,1)

    %Guardo en una nueva matriz de "inter" filas la media de los dos
máximos de las filas de
    %'j-inter' a inter (primer caso por ejemplo de 1 a 10).

    %CÓMO LO HAGO?

    %Selecciono por filas ('x' filas de j-inter+1 hasta j) todos los
    %valores de las columnas. Calculo sobre cuantos máximos tengo que
hacer
    %la media (max10; puedo hacer la media teniendo en cuenta sólo un
10%
    %de los máximos que hay en el intervalo; si hay 10 filas en 1
intervalo
    %y hago el 10%-max10 tendré que hacer la media con sólo 1 fila).

    c=sort(abs(S(j-inter+1:j,:)),1,'descend');
    [a b]=size(c);
    Snew(j/inter,:) = mean(c(1:ceil(inter*max10/100),:),1); % pongo al
final ",1);"
    %para que cuando haya que hacer la media de una sola fila, no me
de un
    %solo resultado, si no que me de uno por cada columna.



    j=j+inter;
end

%Si el último número evaluado, antes de salir del bucle, no era la
última
%frecuencia disponible, tenemos que hacer la media de las filas aun no
%tratadas. (Serán las que vayan de j+1 al máximo del valor de las
frecuencias).
if (j-inter)~=(size(S,1))
    %Para hacer la media hay que tener en cuenta si quedan más o menos
    %filas de las que se precisa para hacer el max-10 %
    if ((inter*max10/100)<(size(S,1)-(j-inter)))
        c=sort(abs(S(j-inter+1:size(S,1),:)),1,'descend');
        Snew(j/inter,:) =
mean(c(ceil(inter*max10/100)+1:size(c,1),:),1);
    %Si hay menos filas que las que necesita max-10 se hace la media
normal
```

```
    else
        Snew(j/inter,:) = mean(S(j-inter+1:size(S,1),:),1);
    end
end
y=Snew;
figure;imagesc(size(Snew,2),size(Snew,1),flipud(20*log10(abs(Snew))));


function [miMatriz fonemas Sred]=reduceAPhonemas(Sred)
%Obtenemos todos los fonemas de la grabación de forma manual.
%Después calculamos cada longitud de los fonemas y los añadimos a la
matriz
%que tenía la información del fonema.
h = getPhonem(Sred,1,12);
sh = getPhonem(Sred,13,17);
ix = getPhonem(Sred,18,22);
hv = getPhonem(Sred,23,26);
eh = getPhonem(Sred,27,34);
dcl = getPhonem(Sred,35,36);
jh = getPhonem(Sred,37,40);
ih = getPhonem(Sred,41,45);
dcl2 = getPhonem(Sred,46,48);
d = getPhonem(Sred,49,49);
ah = getPhonem(Sred,50,57);
klc = getPhonem(Sred,58,61);
k = getPhonem(Sred,62,63);
s = getPhonem(Sred,64,70);
ux = getPhonem(Sred,71,79);
q = getPhonem(Sred,80,82);
en = getPhonem(Sred,83,87);
gcl = getPhonem(Sred,88,89);
g = getPhonem(Sred,90,90);
r = getPhonem(Sred,91,94);
ix2 = getPhonem(Sred,95,99);
s2 = getPhonem(Sred,100,105);
ix3 = getPhonem(Sred,106,109);
w = getPhonem(Sred,110,115);
ao = getPhonem(Sred,116,123);
sh2 = getPhonem(Sred,124,129);
epi = getPhonem(Sred,130,131);
w2 = getPhonem(Sred,132,134);
ao2 = getPhonem(Sred,135,140);
dx = getPhonem(Sred,141,141);
axr = getPhonem(Sred,142,145);
ao3 = getPhonem(Sred,146,153);
l = getPhonem(Sred,154,156);
y = getPhonem(Sred,157,163);
ih2 = getPhonem(Sred,164,168);
axr2 = getPhonem(Sred,169,173);
h2 = getPhonem(Sred,174,181);

%Calculamos la longitud de cada fonema.
mediaH=(30.5+21.34)/2; %h#1 y h#2
mediaSH=(15.49+16.41)/2; %sh y sh2
mediaIX=(11.64+13.37+09.08)/3; %ix,ix2 y ix3
mediaDCL=(04.18+09.83)/2; %dcl y dcl2
mediaIH=(11.80+14.20)/2; %ih y ih2
mediaS=(17.54+15.90)/2; %s y s2
mediaW=(15.96+09.61)/2; %w y w2
mediaAO=(20.59+13.65+20.05)/3; %ao, ao2 y ao3
mediaAXR=(12.30+11.07)/2; %axr y axr2
```

```
%Creo una matriz en la que guardaré todos los fonemas(matriz de 3
%dimensiones)
miMatriz=zeros(size(h,1)+1,2);
contador=0;


%La función aLongitud añade en el fonema una fila más correspondiente
al
%tipo de longitud.

h = aLongitud(h,mediaH);
contador=contador+1;
miMatriz(:,:,contador)=h;

sh = aLongitud(sh,mediaSH);
contador=contador+1;
miMatriz(:,:,contador)=sh;

ix = aLongitud(ix,mediaIX);
contador=contador+1;
miMatriz(:,:,contador)=ix;

hv = aLongitud(hv,09.19);
contador=contador+1;
miMatriz(:,:,contador)=hv;

eh = aLongitud(eh,21.30);
contador=contador+1;
miMatriz(:,:,contador)=eh;

dcl = aLongitud(dcl,mediaDCL);
contador=contador+1;
miMatriz(:,:,contador)=dcl;

jh = aLongitud(jh,11.47);
contador=contador+1;
miMatriz(:,:,contador)=jh;

ih = aLongitud(ih,mediaIH);
contador=contador+1;
miMatriz(:,:,contador)=ih;

d = aLongitud(d,01.40);
contador=contador+1;
miMatriz(:,:,contador)=d;

ah = aLongitud(ah,20.74);
contador=contador+1;
miMatriz(:,:,contador)=ah;

klc = aLongitud(klc,11.56);
contador=contador+1;
miMatriz(:,:,contador)=klc;

k = aLongitud(k,04.64);
contador=contador+1;
```

```
miMatriz(:,:,contador)=k;

s = aLongitud(s,mediaS);
contador=contador+1;
miMatriz(:,:,contador)=s;

ux = aLongitud(ux,23.29);
contador=contador+1;
miMatriz(:,:,contador)=ux;

q = aLongitud(q,07.82);
contador=contador+1;
miMatriz(:,:,contador)=q;

en = aLongitud(en,13.61);
contador=contador+1;
miMatriz(:,:,contador)=en;

gcl = aLongitud(gcl,03.60);
contador=contador+1;
miMatriz(:,:,contador)=gcl;

g = aLongitud(g,03.51);
contador=contador+1;
miMatriz(:,:,contador)=g;

r = aLongitud(r,09.58);
contador=contador+1;
miMatriz(:,:,contador)=r;

w = aLongitud(w,mediaW);
contador=contador+1;
miMatriz(:,:,contador)=w;

ao = aLongitud(ao,mediaAO);
contador=contador+1;
miMatriz(:,:,contador)=ao;

epi = aLongitud(epi,03.94);
contador=contador+1;
miMatriz(:,:,contador)=epi;

dx = aLongitud(dx,02.46);
contador=contador+1;
miMatriz(:,:,contador)=dx;

axr = aLongitud(axr,mediaAXR);
contador=contador+1;
miMatriz(:,:,contador)=axr;

l = aLongitud(l,07.52);
contador=contador+1;
miMatriz(:,:,contador)=l;

y = aLongitud(y,17.46);
contador=contador+1;
miMatriz(:,:,contador)=y;
```

```
fonemas={'h' 'sh' 'ix' 'hv' 'eh' 'dcl' 'jh' 'ih' 'd' 'ah' 'klc' 'k'
's' 'ux' 'q' 'en' 'gcl' 'g' 'r' 'w' 'ao' 'epi' 'dx' 'axr' 'l' 'y'};
```

```
function y=getPhonem(Sred,inic,fin)
%Hacemos la media en la matriz Sred de todas las columnas entre inicio
y
%fin. Después obtenemos los colores para el fonema y nos quedamos con
los
%dos máximos.
y1=mean(Sred(:,inic:fin),2);

%Una vez tenemos el fonema, calculamos sus colores y sus máximos (m1 y
m2)
[a b c d e f g h m1 m2]=aColores(y1);

%Guardamos los dos máximos como parte del fonema.
y(:,1)=m1;
y(:,2)=m2;
```

```
function [matriz fonemas]=haceTodo()
%[black blue purple red yellow green cyan white maximo1 maximo2]
[S f t]=grab2mel(512);
Sred=speechreduction(S,10);
[matriz fonemas grabacion]=reduceAPhonemas(Sred);
```

Funciones del reconocimiento

```
function fonemasRec=reconocimiento(Matriz, grabacion, funciones)
%Realiza el reconocimiento de los fonemas que están en matriz sobre la
%grabación

%Matriz es la matriz 3-Dimensional con cada uno de los fonemas y sus
%máximos, mínimos colores y longitudes.

%grabación es la matriz de la grabación habiendose hecho el paso a mel
y la reducción
%correspondiente de la información. (grab2mel(512) &
%speechreduction3(S,10))

%funciones es la matriz con los valores Start,first,last y end del
calculo
%de los colores y longitudes.

%tiempos es la matriz que indica en que tiempos empieza cada fonema y
el
%número de fonema que encaja en cada parte. Lo usamos para contar el
número
%de aciertos en la búsqueda.
tiempos=[1 1; %h#
         13 2; %sh
         18 3; %ix
         23 4; %hv
```

```
27 5; %eh
35 6; %dcl
37 7; %jh
41 8; %ih
46 6; %dcl
49 9; %d
50 10; %ah
58 11; %kcl
62 12; %k
64 13; %s
71 14; %ux
80 15; %q
83 16; %en
88 17; %gcl
90 18; %g
91 19; %r
95 3; %ix
100 13; %s
106 3; %ix
110 20; %w
116 21; %ao
124 2; %sh
130 22; %epi
132 20; %w
135 21; %ao
141 23; %dx
142 24; %axr
146 21; %ao
154 25; %l
157 26; %y
164 8; %ih
169 24; %axr
174 1]; %h#


%Vamos a reconocer en cada fragmento de la grabación los diferentes
fonemas
%que tenemos y el porcentaje de acierto que hay en cada uno para
guardarlos en fonemasRec.


for i=1:size(tiempos,1)-1

    %resultados es una matriz con los aciertos de cada fonema en este
    %fragmento de la grabación.

resultados=jugarEntrenamiento(Matriz,grabacion,funciones,tiempos(i,1),
tiempos(i+1,1));

    fonemasRec(i,:,1)=resultados(1,:);
    fonemasRec(i,:,2)=resultados(2,:);
    fonemasRec(i,:,3)=resultados(3,:);

end
floor(fonemasRec(:,1,1).*100)
fonemasRec(:,2,1)
floor(fonemasRec(:,1,2).*100)
fonemasRec(:,2,2)
floor(fonemasRec(:,1,3).*100)
fonemasRec(:,2,3)
```

```
function y=jugarEntrenamiento(MatPhonem,S,funciones,inicio,final)
%Esta función busca, a partir de un "inicio" dado en S, alguna
similitud de un
%fonema de la matriz(MatPhonem). Este fonema encontrado se devuelve en
y.

%funciones es la matriz con todos los valores que se aplican en la
función
%generalizada para los colores y longitudes.

%Inicio indica desde que punto(tiempo) queremos seguir buscando en el
%espectrograma.

%El espectrograma obtenido ya está reducido.
espectrograma=S;
y=0;
%Inicializo la y a 0 para indicar que no se ha encontrado ningún
fonema.

%Inicializo la variable que tendrá en cuenta que se puedan encontrar
varios
%resultados válidos en un mismo momento, y la matriz en la que
guardaré los resultados.
contRes=1;
resultados=zeros(1,2);


%Definimos las variables Umbral.
minParaContar=0.60;
minParaValidar=75;

%Para empezar recorremos todo el espectrograma columna a columna
porque
%queremos tratar cada una individualmente.



if final==0 % SI NO SE HA METIDO UN NUMERO COMO FINAL, LO REALIZO A
TODA LA GRABACIÓN Y DEVUELVO EL FONEMA QUE ENCUENTRA

    %Antes vamos a crear una matriz de zeros de tantos fonemas como
tenemos por
    %filas, y tantas columnas como tiene el espectrograma (+1 en la
que guardaremos la longitud más larga
    %de valores>60 encontrados), para guardar en ella
    %los resultados que obtenemos para cada columna con cada fonema.
    minValue=zeros(size(MatPhonem,2)/2,size(espectrograma,2));




    %Mientras en la matriz resultados haya zeros y no hayamos llegado
al final, querrá decir que aun no se ha
    %encontrado nada, así que seguimos mirando o salimos.
    i=inicio;
    while (y==0 && i<=size(espectrograma,2))
```

```
        %Creamos las 8 matrices de colores para cada frame.
        %A partir de las 8 matrices creamos la matriz de pertenencia.
        mColores=zeros(size(espectrograma,1),8);
        for j=1:8

mColores(:,j)=coloresTipoEntrenamiento(j,espectrograma(:,i),funciones)
;
        end

        pertenencia=MatPhonem(1:size(MatPhonem,1)-1,:); %Pertenencia
tiene todos los fonemas menos la longitud.
        %Creo la matriz de pertenencia a partir de las 8 matrices de
colores
        %(fila a fila).
        for k=1:size(pertenencia,1) %Como el espectrograma tiene 25
bandas este bucle solo se hace 25 veces.
            pertenencia(k,pertenencia(k,:)==1)=mColores(k,1);
            pertenencia(k,pertenencia(k,:)==2)=mColores(k,2);
            pertenencia(k,pertenencia(k,:)==3)=mColores(k,3);
            pertenencia(k,pertenencia(k,:)==4)=mColores(k,4);
            pertenencia(k,pertenencia(k,:)==5)=mColores(k,5);
            pertenencia(k,pertenencia(k,:)==6)=mColores(k,6);
            pertenencia(k,pertenencia(k,:)==7)=mColores(k,7);
            pertenencia(k,pertenencia(k,:)==8)=mColores(k,8);
        end

        %Creo dos matrices, una con los colores 1 de cada fonema y
otra con los
        %colores 2 de cada fonema. Hago esto para luego coger el
máximo de las
        %dos.

        m1=pertenencia(:,1:2:size(pertenencia,2));
        m2=pertenencia(:,2:2:size(pertenencia,2));

        %Obtengo los mínimos valores de los máximos. Hago la
traspuesta para
        %guardar los mínimos por columnas.

        minValue(:,i)=min(max(m1,m2))';

        %Inicializo la variable longitudes con tantas filas como
fonemas y una
        %sola columna.
        longitudes=zeros(size(minValue,1),1);


        %Aumento la longitude de valores > que el umbral en 1 o pongo
a cero
        %las que encuentro por debajo.

longitudes(minValue(:,i)>=minParaContar)=longitudes(minValue(:,i)>minP
araContar)+1;
        longitudes(minValue(:,i)<minParaContar)=0;

        %Comprobamos para cada fonema si cumple que se ha reconocido.
        for k=1:size(minValue,1) %el número de fonemas (26)
```

```
            %Voy a coger todos los valores consecutivos > que 60 para
hacer su
            %media. El número de valores consecutivos me indica la
posible
            %longitud del fonema. Aplico la función tipoLongitud() a
ésta(donde
            %el primer parámetro de la función será la longitud del
fonema con
            %el que lo estamos comparando) y lo que me devuelva lo
multiplico
            %con la media de valores>60.
            if longitudes(k) ~= 0
                inicioMedia=i-longitudes(k,1)+1; %la columna en la que
estamos - los valores que llevamos consecutivos.
                finMedia=i;%la columna hasta la que hacemos la media


media=mean(minValue(k,inicioMedia:finMedia))*100;%media de los valores
entre inicio y fin. Lo paso a valores entre 0 y 100 en lugar
                %de 0 a 1.

                %tipoLongitud() recibe como primer parámetro el tipo
de
                %longitud(vs,s,vl...) que se encuentra en la última
fila, columna 1
                %de cada fonema (MatPhonem), y la longitud con la que
estamos trabajando.

fonemLeng=longitudTipoEntrenamiento(MatPhonem(size(MatPhonem,1),k*2-
1),longitudes(k),funciones);

                %El resultado final de la evaluación del fonema es
resultadoFinal.
                resultadoFinal=media*fonemLeng;

                %Si el resultado final pasa el umbral establecido.
                if resultadoFinal>=minParaValidar

                    %Guardamos el fonema (número que identifica un
fonema) en la
                    %matriz de resultados.
                    resultados(contRes,1)=k;

                    %Guardamos el porcentaje de parecido
(resultadoFinal) que tiene
                    %el fonema encontrado con el comparado.
                    resultados(contRes,2)=resultadoFinal;

                    %Aumentamos el contador para que se guarde la
siguiente.
                    contRes=contRes+1;

                end
            end
        end

        %Si la matriz de resultados tiene en la posición 1 todavía un
0, quiere
        %decir que no ha encontrado ningún resultado válido.Sin
embargo, si ha
```

```
        %encontrado algún resultado, coge el fonema que tenga mayor
proximidad (columna
        %2) y lo guarda en 'y'.
        if resultados(1,1)~=0 %Si ha encontrado algún fonema será como
mínimo el fonema 1.

            % cojo el % con mayor exactitud.
            porcentajeMaximo=max(resultados(:,2));

            %busco en qué posición de la matriz resultados se
encuentra.

fonema=resultados(find(resultados(:,2)==porcentajeMaximo));

            %En la matriz resultados, la posición 'fonema' se
encuentra el
            %verdadero identificador del fonema que queremos guardar.
            y=fonema(1);

        end
        i=i+1;
    end
else
    if final > 0 %SI SE HA METIDO UN VALOR DESDE EL QUE ACABAR, LA
FUNCIÓN NOS DEVOLVERÁ LA TABLA CON LOS RESULTADOS DE CADA FONEMA

        %Creo la matriz minValue que va a tener un tamaño muy
específico en
        %este caso (26 fonemas, los frames que van desde inicio hasta
final ambos incluidos).
        minValue=zeros(size(MatPhonem,2)/2,final);

        %Variable que indica con que porcentaje de valores de minvalue
se
        %hace la media.
        Average=80;

        %Bucle que recorre todos los frames desde inicio hasta final.
        for i=inicio:final

            %Para cada frame voy a ir creando las 8 matrices de
colores para
            %calcular cada fila de minValue.
            mColores=zeros(size(espectrograma,1),8);
            for j=1:8

mColores(:,j)=coloresTipoEntrenamiento(j,espectrograma(:,i),funciones)
;
            end

            pertenencia=MatPhonem(1:size(MatPhonem,1)-1,:);
%Pertenencia tiene todos los fonemas menos la longitud.
            %Creo la matriz de pertenencia a partir de las 8 matrices
de colores
            %(fila a fila).
            for k=1:size(pertenencia,1) %Como el espectrograma tiene
25 bandas este bucle solo se hace 25 veces.
                pertenencia(k,pertenencia(k,:)==1)=mColores(k,1);
                pertenencia(k,pertenencia(k,:)==2)=mColores(k,2);
                pertenencia(k,pertenencia(k,:)==3)=mColores(k,3);
```

```
                pertenencia(k,pertenencia(k,:)==4)=mColores(k,4);
                pertenencia(k,pertenencia(k,:)==5)=mColores(k,5);
                pertenencia(k,pertenencia(k,:)==6)=mColores(k,6);
                pertenencia(k,pertenencia(k,:)==7)=mColores(k,7);
                pertenencia(k,pertenencia(k,:)==8)=mColores(k,8);
            end

            %Creo dos matrices, una con los colores 1 de cada fonema y
otra con los
            %colores 2 de cada fonema. Hago esto para luego coger el
máximo de las
            %dos.

            m1=pertenencia(:,1:2:size(pertenencia,2));
            m2=pertenencia(:,2:2:size(pertenencia,2));

            %Obtengo los mínimos valores de los máximos. Hago la
traspuesta para
            %guardar los mínimos por columnas.

            minValue(:,i)=min(max(m1,m2))';
        end
       % minValue

        %Guardamos todas las longitudes de cada fonema.

resLongitudes=MatPhonem(size(MatPhonem,1),1:2:size(MatPhonem,2))';
       % resLongitudes

        %ya tenemos minValue con toda la información. Procedemos a
hacer el
        %AverageN
        resultados=sort(minValue(:,inicio:final),2,'descend');

resultados=mean(resultados(:,1:ceil(Average*size(resultados,2)/100)),2
);

       % resultados

        %Calculamos la pertenencia de la longitud del bloque en cada
una de
        %las longitudes posibles
        for j=1:5
            aLongitudes(j)=longitudTipoEntrenamiento(j,final-
inicio+1,funciones);
        end

        resLongitudes(resLongitudes==1)=aLongitudes(1);
        resLongitudes(resLongitudes==2)=aLongitudes(2);
        resLongitudes(resLongitudes==3)=aLongitudes(3);
        resLongitudes(resLongitudes==4)=aLongitudes(4);
        resLongitudes(resLongitudes==5)=aLongitudes(5);

      %  resLongitudes

        %Multiplico la media de minValue por el grado de pertenencia
de la
        %longitud de cada fonema para ajustar y calcular el valor de
        %coincidencia.
```

```
        resultados=resultados.*resLongitudes;

        %Hago un bucle de forma que trato todos los fonemas.
        for k=1:size(resultados,1)
            unNumero=find(resultados==max(resultados));

            %En max(resultados) tengo el valor de coincidencia del
fonema.
            %En unNumero tengo la posición donde se encontraba ese
valor, y
            %que corresponde al fonema.
            res(k,1)=max(resultados);
            res(k,2)=unNumero(1);

            %Pongo el número máximo a -1.
            resultados(unNumero(1))=-1;
        end

        %Devolvemos la tabla de resultados.
        y=res;

    else
            x='Error, final es un número negativo';
            x
    end
end
```

```
function fonemasRec=reconocimientoDatos(Matriz, grabacion, funciones)
%Realiza el reconocimiento de los fonemas de la matriz sobre las dos
%grabaciones que se obtienen al reducir grabación

%Matriz es la matriz 3-Dimensional con cada uno de los fonemas y sus
%máximos, mínimos colores y longitudes.

%grabación es la muestra sin reducir.
[minimos maximos]=reduccionIntervalos(grabacion);

%funciones es la matriz con los valores Start,first,last y end del
calculo
%de los colores y longitudes.

%tiempos es la matriz que indica en que tiempos empieza cada fonema y
el
%número de fonema que encaja en cada parte. Lo usamos para contar el
número
%de aciertos en la búsqueda.
tiempos=[1 1; %h#
         13 2; %sh
         18 3; %ix
         23 4; %hv
         27 5; %eh
         35 6; %dcl
         37 7; %jh
         41 8; %ih
```

```
46 6; %dcl
49 9; %d
50 10; %ah
58 11; %kcl
62 12; %k
64 13; %s
71 14; %ux
80 15; %q
83 16; %en
88 17; %gcl
90 18; %g
91 19; %r
95 3; %ix
100 13; %s
106 3; %ix
110 20; %w
116 21; %ao
124 2; %sh
130 22; %epi
132 20; %w
135 21; %ao
141 23; %dx
142 24; %axr
146 21; %ao
154 25; %l
157 26; %y
164 8; %ih
169 24; %axr
174 1]; %h#


%Vamos a reconocer en cada fragmento de la grabación los diferentes
fonemas
%que tenemos y el porcentaje de acierto que hay en cada uno para
guardarlos en fonemasRec.


for i=1:size(tiempos,1)-1

    %resultados es una matriz con los aciertos de cada fonema en este
    %fragmento de la grabación.

resultados=jugarIntervalosDatos(Matriz,minimos,maximos,funciones,tiemp
os(i,1),tiempos(i+1,1));

    fonemasRec(i,:,1)=resultados(1,:);
    fonemasRec(i,:,2)=resultados(2,:);
    fonemasRec(i,:,3)=resultados(3,:);

end
floor(fonemasRec(:,1,1).*100)
fonemasRec(:,2,1)
floor(fonemasRec(:,1,2).*100)
fonemasRec(:,2,2)
floor(fonemasRec(:,1,3).*100)
fonemasRec(:,2,3)
```

```
function [minimos maximos]=reduccionIntervalos(S)
%devuelve dos matrices reducidas. Una con los mínimos de cada banda y
otra
%con los máximos.

if S==0

    %Hacemos el spectrograma y guardamos las frecuencias FFT y los
ejes X e Y
    %(tiempo y frecuencias)
    [S f t]=grab2mel(512);
end
%Guardo el intervalo con el que voy a generar las 25 bandas.
inter=floor(size(S,1)/25);

%'j' va a comenzar en 1 + "intervalo". Queremos coger la primera vez
%desde 1 hasta j para coger la mínima y la máxima.
j=inter;
while j<=size(S,1)

    %Como me interesa coger el máximo y el mínimo, ordeno todos los
valores
    %de cada columna para coger el primero y el último.
    minimos(j/inter,:)=min(sort(abs(S(j-inter+1:j,:)),1,'descend'));
    maximos(j/inter,:)=max(sort(abs(S(j-inter+1:j,:)),1,'descend'));



    j=j+inter;
end

%Si el último número evaluado, antes de salir del bucle, no era la
última
%frecuencia disponible, tenemos que tratar los valores hasta el final.
if (j-inter)~=(size(S,1))

        minimos(j/inter,:)=min(sort(abs(S(j-
inter+1:size(S,1),:)),1,'descend'));
        maximos(j/inter,:)=max(sort(abs(S(j-
inter+1:size(S,1),:)),1,'descend'));

end




function
y=jugarIntervalosDatos(MatPhonem,minimos,maximos,funciones,inicio,fina
l)
%Esta función busca, a partir de un "inicio" dado en S, alguna
similitud de un
%fonema de la matriz(MatPhonem). Cuando termina devuelve todos los
fonemas
%con su % de similitud.
```

```
%funciones es la matriz con todos los valores que se aplican en la
función
%generalizada para los colores y longitudes.

%Inicio indica desde que punto(tiempo) queremos seguir buscando en el
%espectrograma.

%El espectrograma obtenido ya está reducido. Obtenemos una grabación
de
%minimos y otra de máximos.
espectrogramaMin=minimos;
espectrogramaMax=maximos;

% Inicializo la matriz en la que guardo los resultados.
resultados=zeros(1,2);


%Definimos las variables Umbral.
minParaContar=[0.40 0.60] ;
minParaValidar=75;



if final==0 % SI NO SE HA METIDO UN NUMERO COMO FINAL, LO REALIZO A
TODA LA GRABACIÓN Y DEVUELVO EL FONEMA QUE ENCUENTRA



    %Antes vamos a crear una matriz de zeros de tantos fonemas como
tenemos por
    %filas, y tantas columnas como tiene el espectrograma (+1 en la
que guardaremos la longitud más larga
    %de valores>60 encontrados), para guardar en ella
    %los resultados que obtenemos para cada columna con cada fonema.
    %Creamos dos matrices para almacenar el valor "min" mínimo del
    %intervalo y el valor "max" mínimo del intervalo.
    minValueMin=zeros(size(MatPhonem,2)/2,size(espectrogramaMin,2));
    minValueMax=zeros(size(MatPhonem,2)/2,size(espectrogramaMax,2));

    %Inicializo la y a 0 para indicar que no se ha encontrado ningún
fonema.
    y=0;

    %Inicializo la variable que tendrá en cuenta que se puedan
encontrar varios
    %resultados válidos en un mismo momento.
    contRes=1;



    %Mientras en la matriz resultados haya zeros y no hayamos llegado
al final, querrá decir que aun no se ha
    %encontrado nada, así que seguimos mirando o salimos.
    i=inicio;
    while (y==0 && i<=size(espectrogramaMin,2))


        %Creamos las 8 matrices de colores para cada frame.
```

```
        %A partir de las 8 matrices creamos la matriz de pertenencia.
        %Creamos 8 para los mínimos y 8 para los máximos




        %PARA LOS MINIMOS
        mColores=zeros(size(espectrogramaMin,1),8);
        for j=1:8

mColores(:,j)=coloresTipoEntrenamiento(j,espectrogramaMin(:,i),funcion
es);
        end

        pertenencia=MatPhonem(1:size(MatPhonem,1)-1,:); %Pertenencia
tiene todos los fonemas menos la longitud.
        %Creo la matriz de pertenencia para la grabación con los
valoes mínimos a partir de las 8 matrices de colores
        %(fila a fila) que hemos creado del "espectrograma Min".
        for k=1:size(pertenencia,1) %Como el espectrograma tiene 25
bandas este bucle solo se hace 25 veces.
            pertenencia(k,pertenencia(k,:)==1)=mColores(k,1);
            pertenencia(k,pertenencia(k,:)==2)=mColores(k,2);
            pertenencia(k,pertenencia(k,:)==3)=mColores(k,3);
            pertenencia(k,pertenencia(k,:)==4)=mColores(k,4);
            pertenencia(k,pertenencia(k,:)==5)=mColores(k,5);
            pertenencia(k,pertenencia(k,:)==6)=mColores(k,6);
            pertenencia(k,pertenencia(k,:)==7)=mColores(k,7);
            pertenencia(k,pertenencia(k,:)==8)=mColores(k,8);
        end

        %Creo dos matrices, una con los colores 1 de cada fonema y
otra con los
        %colores 2 de cada fonema. Hago esto para luego coger el
máximo de las
        %dos.

        m1=pertenencia(:,1:2:size(pertenencia,2));
        m2=pertenencia(:,2:2:size(pertenencia,2));

        %Obtengo los mínimos valores de los máximos. Hago la
traspuesta para
        %guardar los mínimos por columnas.

        minValueMin(:,i)=min(max(m1,m2))';




        %PARA LOS MAXIMOS
        mColores=zeros(size(espectrogramaMax,1),8);
        for j=1:8
```

```
mColores(:,j)=coloresTipoEntrenamiento(j,espectrogramaMax(:,i),funcion
es);
        end

        pertenencia=MatPhonem(1:size(MatPhonem,1)-1,:); %Pertenencia
tiene todos los fonemas menos la longitud.
        %Creo la matriz de pertenencia para la grabación con los
valoes mínimos a partir de las 8 matrices de colores
        %(fila a fila) que hemos creado del "espectrograma Min".
        for k=1:size(pertenencia,1) %Como el espectrograma tiene 25
bandas este bucle solo se hace 25 veces.
            pertenencia(k,pertenencia(k,:)==1)=mColores(k,1);
            pertenencia(k,pertenencia(k,:)==2)=mColores(k,2);
            pertenencia(k,pertenencia(k,:)==3)=mColores(k,3);
            pertenencia(k,pertenencia(k,:)==4)=mColores(k,4);
            pertenencia(k,pertenencia(k,:)==5)=mColores(k,5);
            pertenencia(k,pertenencia(k,:)==6)=mColores(k,6);
            pertenencia(k,pertenencia(k,:)==7)=mColores(k,7);
            pertenencia(k,pertenencia(k,:)==8)=mColores(k,8);
        end

        %Creo dos matrices, una con los colores 1 de cada fonema y
otra con los
        %colores 2 de cada fonema. Hago esto para luego coger el
máximo de las
        %dos.

        m1=pertenencia(:,1:2:size(pertenencia,2));
        m2=pertenencia(:,2:2:size(pertenencia,2));

        %Obtengo los mínimos valores de los máximos. Hago la
traspuesta para
        %guardar los mínimos por columnas.

        minValueMax(:,i)=min(max(m1,m2))';



        %Inicializo la variable longitudes con tantas filas como
fonemas y una
        %sola columna. Uso el tamaño de minValueMax como podría usar
el de
        %minValueMin.
        longitudes=zeros(size(minValueMax,1),1);


        %Aumento la longitude de valores > que el umbral en 1 o pongo
a cero
        %las que encuentro por debajo.
        longitudes(minValueMin(:,i) >= minParaContar(1)  &
minValueMax(:,i) >=
minParaContar(2))=longitudes(minValue(:,i)>minParaContar)+1;
        longitudes(minValueMin(:,i) < minParaContar(1)  |
minValueMax(:,i) < minParaContar(2))=0;

        %Comprobamos para cada fonema si cumple que se ha reconocido.
        %Uso el tamaño de minValueMax como podría usar el de
        %minValueMin.
```

```
        for k=1:size(minValueMax,1) %el número de fonemas (26)

            %Voy a coger todos los valores consecutivos con min>=50 y
max>=60 para hacer una media
            %en minValueMin y minValueMax, y obener un valor mínimo y
máximo medio del intervalo.
            %El número de valores consecutivos me indica la posible
            %longitud del fonema. Voy a obtener un parámetro de
            %coincidencia de la longitud encontrada con la del fonema.
            %Dependiendo de este valor, se ajustará al valor mínimo o
al
            %máximo del intervalo para mostrarlo como resultado final.
            if longitudes(k) ~= 0
                inicioMedia=i-longitudes(k,1)+1; %la columna en la que
estamos - los valores que llevamos consecutivos.
                finMedia=i;%la columna hasta la que hacemos la media


mediaMinima=mean(minValueMin(k,inicioMedia:finMedia))*100;%media de
los valores entre inicio y fin. Lo paso a valores entre 0 y 100 en
lugar
                %de 0 a 1.

mediaMaxima=mean(minValueMax(k,inicioMedia:finMedia))*100;

                %tipoLongitud() recibe como primer parámetro el tipo
de
                %longitud(vs,s,vl...) que se encuentra en la última
fila, columna 1
                %de cada fonema (MatPhonem), y la longitud con la que
estamos trabajando.

fonemLeng=longitudTipoEntrenamiento(MatPhonem(size(MatPhonem,1),k*2-
1),longitudes(k),funciones);

                %El resultado final de la evaluación del fonema es
resultadoFinal.
                %Se trata de un ajuste del intervalo al valor mínimo o
                %máximo dependiendo de la pertenencia del fonema a esa
                %longitud.
                resultadoFinal=mediaMinima+(fonemLeng*(mediaMaxima-
mediaMinima));

                %Si el resultado final pasa el umbral establecido.
                if resultadoFinal>=minParaValidar

                    %Guardamos el fonema (número que identifica un
fonema) en la
                    %matriz de resultados.
                    resultados(contRes,1)=k;

                    %Guardamos el porcentaje de parecido
(resultadoFinal) que tiene
                    %el fonema encontrado con el comparado.
                    resultados(contRes,2)=resultadoFinal;

                    %Aumentamos el contador para que se guarde la
siguiente.
                    contRes=contRes+1;
```

```
                end
            end
        end

        %Si la matriz de resultados tiene en la posición 1 todavía un
0, quiere
        %decir que no ha encontrado ningún resultado válido.Sin
embargo, si ha
        %encontrado algún resultado, coge el fonema que tenga mayor
proximidad (columna
        %2) y lo guarda en 'y'.
        if resultados(1,1)~=0 %Si ha encontrado algún fonema será como
mínimo el fonema 1.

            % cojo el % con mayor exactitud.
            porcentajeMaximo=max(resultados(:,2));

            %busco en qué posición de la matriz resultados se
encuentra.

fonema=resultados(find(resultados(:,2)==porcentajeMaximo));

            %En la matriz resultados, la posición 'fonema' se
encuentra el
            %verdadero identificador del fonema que queremos guardar.
            y=fonema(1);

        end
        i=i+1;
    end
else
    if final > 0 %SI SE HA METIDO UN VALOR DESDE EL QUE ACABAR, LA
FUNCIÓN NOS DEVOLVERÁ LA TABLA CON LOS RESULTADOS DE CADA FONEMA


        %Creo la matriz minValue que va a tener un tamaño muy
específico en
        %este caso (26 fonemas, los frames que van desde inicio hasta
final ambos incluidos).
        minValueMin=zeros(size(MatPhonem,2)/2,final);
        minValueMax=zeros(size(MatPhonem,2)/2,final);


        %Variable que indica con que porcentaje de valores de minvalue
se
        %hace la media.
        Average=80;

        %Bucle que recorre todos los frames desde inicio hasta final.
        for i=inicio:final


            %PARA LOS MINIMOS


            %Para cada frame voy a ir creando las 8 matrices de
colores para
            %calcular cada fila de minValue.
            mColores=zeros(size(espectrogramaMin,1),8);
```

```
            for j=1:8

mColores(:,j)=coloresTipoEntrenamiento(j,espectrogramaMin(:,i),funcion
es);
            end

            pertenencia=MatPhonem(1:size(MatPhonem,1)-1,:);
%Pertenencia tiene todos los fonemas menos la longitud.
            %Creo la matriz de pertenencia a partir de las 8 matrices
de colores
            %(fila a fila).
            for k=1:size(pertenencia,1) %Como el espectrograma tiene
25 bandas este bucle solo se hace 25 veces.
                pertenencia(k,pertenencia(k,:)==1)=mColores(k,1);
                pertenencia(k,pertenencia(k,:)==2)=mColores(k,2);
                pertenencia(k,pertenencia(k,:)==3)=mColores(k,3);
                pertenencia(k,pertenencia(k,:)==4)=mColores(k,4);
                pertenencia(k,pertenencia(k,:)==5)=mColores(k,5);
                pertenencia(k,pertenencia(k,:)==6)=mColores(k,6);
                pertenencia(k,pertenencia(k,:)==7)=mColores(k,7);
                pertenencia(k,pertenencia(k,:)==8)=mColores(k,8);
            end

            %Creo dos matrices, una con los colores 1 de cada fonema y
otra con los
            %colores 2 de cada fonema. Hago esto para luego coger el
máximo de las
            %dos.

            m1=pertenencia(:,1:2:size(pertenencia,2));
            m2=pertenencia(:,2:2:size(pertenencia,2));

            %Obtengo los mínimos valores de los máximos. Hago la
traspuesta para
            %guardar los mínimos por columnas.

            minValueMin(:,i)=min(max(m1,m2))'; %hago i-inicio+1 para
que se guarde en la posición
            %1,2,3,4 y no en la 7,8,9, ya que cuando haga mean se
realizará
            %de todos los elementos de la fila. Si empieza en el 20,
se
            %hará la media de lo que sea y 19 ceros (1-19).



            %PARA LOS MAXIMOS


            %Para cada frame voy a ir creando las 8 matrices de
colores para
            %calcular cada fila de minValue.
            mColores=zeros(size(espectrogramaMax,1),8);
            for j=1:8

mColores(:,j)=coloresTipoEntrenamiento(j,espectrogramaMax(:,i),funcion
es);
            end
```

```
            pertenencia=MatPhonem(1:size(MatPhonem,1)-1,:);
%Pertenencia tiene todos los fonemas menos la longitud.
            %Creo la matriz de pertenencia a partir de las 8 matrices
de colores
            %(fila a fila).
            for k=1:size(pertenencia,1) %Como el espectrograma tiene
25 bandas este bucle solo se hace 25 veces.
                pertenencia(k,pertenencia(k,:)==1)=mColores(k,1);
                pertenencia(k,pertenencia(k,:)==2)=mColores(k,2);
                pertenencia(k,pertenencia(k,:)==3)=mColores(k,3);
                pertenencia(k,pertenencia(k,:)==4)=mColores(k,4);
                pertenencia(k,pertenencia(k,:)==5)=mColores(k,5);
                pertenencia(k,pertenencia(k,:)==6)=mColores(k,6);
                pertenencia(k,pertenencia(k,:)==7)=mColores(k,7);
                pertenencia(k,pertenencia(k,:)==8)=mColores(k,8);
            end

            %Creo dos matrices, una con los colores 1 de cada fonema y
otra con los
            %colores 2 de cada fonema. Hago esto para luego coger el
máximo de las
            %dos.

            m1=pertenencia(:,1:2:size(pertenencia,2));
            m2=pertenencia(:,2:2:size(pertenencia,2));

            %Obtengo los mínimos valores de los máximos. Hago la
traspuesta para
            %guardar los mínimos por columnas.

            minValueMax(:,i)=min(max(m1,m2))'; %hago i-inicio+1 para
que se guarde en la posición
            %1,2,3,4 y no en la 7,8,9, ya que cuando haga mean se
realizará
            %de todos los elementos de la fila. Si empieza en el 20,
se
            %hará la media de lo que sea y 19 ceros (1-19).

        end


        %PARA LONGITUD

        %Guardamos todas las longitudes de cada fonema.

resLongitudes=MatPhonem(size(MatPhonem,1),1:2:size(MatPhonem,2))';

        %ya tenemos minValue con toda la información. Procedemos a
hacer el
        %AverageN sobre los valores mínimos y los máximos
        resultadosMin=sort(minValueMin(:,inicio:final),2,'descend');

resultadosMin=mean(resultadosMin(:,1:ceil(Average*size(resultadosMin,2
)/100)),2);

        resultadosMax=sort(minValueMax(:,inicio:final),2,'descend');

resultadosMax=mean(resultadosMax(:,1:ceil(Average*size(resultadosMax,2
)/100)),2);
```

```
        %Calculamos la pertenencia de la longitud del bloque en cada
una de
        %las longitudes posibles
        for j=1:5
            aLongitudes(j)=longitudTipoEntrenamiento(j,final-
inicio+1,funciones);
        end

        resLongitudes(resLongitudes==1)=aLongitudes(1);
        resLongitudes(resLongitudes==2)=aLongitudes(2);
        resLongitudes(resLongitudes==3)=aLongitudes(3);
        resLongitudes(resLongitudes==4)=aLongitudes(4);
        resLongitudes(resLongitudes==5)=aLongitudes(5);

        %Calculamos el valor final del intervalo ajustándolo con el
valor
        %de la pertenencia de la longitud que tiene cada fonema.
        resultadoFinal=resultadosMin+(resLongitudes.*(resultadosMax-
resultadosMin));


        %Hago un bucle de forma que trato todos los fonemas.
        for k=1:size(resultadoFinal,1)
            unNumero=find(resultadoFinal==max(resultadoFinal));

            %En max(MediaFinal) tengo el valor de coincidencia del
fonema.
            %En unNumero tengo la posición donde se encontraba ese
valor, y
            %que corresponde al fonema.
            res(k,1)=max(resultadoFinal);
            res(k,2)=unNumero(1);

            %Pongo el número máximo a -1.
            resultadoFinal(unNumero(1))=-1;
        end

        %Devolvemos la tabla de resultados.
        y=res;


    else
            x='Error, final es un número negativo';
            x
    end
end
```

```
function fonemasRec=reconocimientoFunciones(Matriz, grabacion,
funciones)
%Reconoce los fonemas de Matriz en la muestra grabación

%Matriz es la matriz 3-Dimensional con cada uno de los fonemas y sus
%máximos, mínimos colores y longitudes.

%grabación es la muestra ya transformada y reducida.

%funciones es la matriz con los valores Start,first,last y end del
calculo
%de los colores y longitudes.

%tiempos es la matriz que indica en que tiempos empieza cada fonema y
el
%número de fonema que encaja en cada parte. Lo usamos para contar el
número
%de aciertos en la búsqueda.
tiempos=[1 1; %h#
         13 2; %sh
         18 3; %ix
         23 4; %hv
         27 5; %eh
         35 6; %dcl
         37 7; %jh
         41 8; %ih
         46 6; %dcl
         49 9; %d
         50 10; %ah
         58 11; %kcl
         62 12; %k
         64 13; %s
         71 14; %ux
         80 15; %q
         83 16; %en
         88 17; %gcl
         90 18; %g
         91 19; %r
         95 3; %ix
         100 13; %s
         106 3; %ix
         110 20; %w
         116 21; %ao
         124 2; %sh
         130 22; %epi
         132 20; %w
         135 21; %ao
         141 23; %dx
         142 24; %axr
         146 21; %ao
         154 25; %l
         157 26; %y
         164 8; %ih
         169 24; %axr
         174 1]; %h#


%Vamos a reconocer en cada fragmento de la grabación los diferentes
fonemas
```

```
%que tenemos y el porcentaje de acierto que hay en cada uno para
guardarlos en fonemasRec.


for i=1:size(tiempos,1)-1

    %resultados es una matriz con los aciertos de cada fonema en este
    %fragmento de la grabación.

resultados=jugarIntervalosFunciones(Matriz,grabacion,funciones,tiempos
(i,1),tiempos(i+1,1));

    fonemasRec(i,:,1)=resultados(1,:);
    fonemasRec(i,:,2)=resultados(2,:);
    fonemasRec(i,:,3)=resultados(3,:);

end
floor(fonemasRec(:,1,1).*100)
fonemasRec(:,2,1)
floor(fonemasRec(:,1,2).*100)
fonemasRec(:,2,2)
floor(fonemasRec(:,1,3).*100)
fonemasRec(:,2,3)




function
y=jugarIntervalosFunciones(MatPhonem,grabacion,funciones,inicio,final)
%Esta función busca, a partir de un "inicio" dado en S, alguna
similitud de un
%fonema de la matriz(MatPhonem). Cuando termina devuelve todos los
fonemas
%con su % de similitud.

%funciones es la matriz con todos los valores que se aplican en la
función
%generalizada para los colores y longitudes.

%Inicio indica desde que punto(tiempo) queremos seguir buscando en el
%espectrograma.

%El espectrograma obtenido ya está reducido.
espectrograma=grabacion;


% Inicializo la matriz en la que guardo los resultados.
resultados=zeros(1,2);


%Definimos las variables Umbral.
minParaContar=[0.60 1] ;
minParaValidar=85;



if final==0 % SI NO SE HA METIDO UN NUMERO COMO FINAL, LO REALIZO A
TODA LA GRABACIÓN Y DEVUELVO EL FONEMA QUE ENCUENTRA
```

```
    %Antes vamos a crear una matriz de zeros de tantos fonemas como
tenemos por
    %filas, y tantas columnas como tiene el espectrograma (+1 en la
que guardaremos la longitud más larga
    %de valores>60 encontrados), para guardar en ella
    %los resultados que obtenemos para cada columna con cada fonema.
    %Creamos dos matrices para almacenar el valor "min" mínimo del
    %intervalo y el valor "max" mínimo del intervalo.
    minValueMin=zeros(size(MatPhonem,2)/2,size(espectrograma,2));
    minValueMax=zeros(size(MatPhonem,2)/2,size(espectrograma,2));

    %Inicializo la y a 0 para indicar que no se ha encontrado ningún
fonema.
    y=0;

    %Inicializo la variable que tendrá en cuenta que se puedan
encontrar varios
    %resultados válidos en un mismo momento.
    contRes=1;



    %Mientras en la matriz resultados haya zeros y no hayamos llegado
al final, querrá decir que aun no se ha
    %encontrado nada, así que seguimos mirando o salimos.
    i=inicio;
    while (y==0 && i<=size(espectrograma,2))


        %Creamos las 16 matrices de colores para cada frame (8 de las
funciones máximo y otros 8 de funciones mínimo).
        %A partir de las matrices creamos la matriz de pertenencia.

        mColores=zeros(size(espectrograma,1),8*2);
        for j=1:8

mColores(:,j)=coloresTipoEntrenamiento(j,espectrograma(:,i),funciones)
;

mColores(:,j+8)=coloresTipoEntrenamiento(j+13,espectrograma(:,i),funci
ones);
        end

        pertenenciaMax=MatPhonem(1:size(MatPhonem,1)-1,:);
%Pertenencia tiene todos los fonemas menos la longitud.
        pertenenciaMin=MatPhonem(1:size(MatPhonem,1)-1,:);

        %Creo la matriz de pertenencia para la grabación con los
valores mínimos a partir de las 16 matrices de colores
        %(fila a fila) que hemos creado del "espectrograma Min".
        for k=1:size(pertenenciaMax,1) %Como el espectrograma tiene 25
bandas este bucle solo se hace 25 veces.
            pertenenciaMax(k,pertenenciaMax(k,:)==1)=mColores(k,1);
            pertenenciaMin(k,pertenenciaMin(k,:)==1)=mColores(k,9);

            pertenenciaMax(k,pertenenciaMax(k,:)==2)=mColores(k,2);
```

```
            pertenenciaMin(k,pertenenciaMin(k,:)==2)=mColores(k,10);

            pertenenciaMax(k,pertenenciaMax(k,:)==3)=mColores(k,3);
            pertenenciaMin(k,pertenenciaMin(k,:)==3)=mColores(k,11);

            pertenenciaMax(k,pertenenciaMax(k,:)==4)=mColores(k,4);
            pertenenciaMin(k,pertenenciaMin(k,:)==4)=mColores(k,12);

            pertenenciaMax(k,pertenenciaMax(k,:)==5)=mColores(k,5);
            pertenenciaMin(k,pertenenciaMin(k,:)==5)=mColores(k,13);

            pertenenciaMax(k,pertenenciaMax(k,:)==6)=mColores(k,6);
            pertenenciaMin(k,pertenenciaMin(k,:)==6)=mColores(k,14);

            pertenenciaMax(k,pertenenciaMax(k,:)==7)=mColores(k,7);
            pertenenciaMin(k,pertenenciaMin(k,:)==7)=mColores(k,15);

            pertenenciaMax(k,pertenenciaMax(k,:)==8)=mColores(k,8);
            pertenenciaMin(k,pertenenciaMin(k,:)==8)=mColores(k,16);

        end

        %Creo dos matrices, una con los colores 1 de cada fonema y
otra con los
        %colores 2 de cada fonema. Hago esto para luego coger el
máximo de las
        %dos.

        m1=pertenenciaMax(:,1:2:size(pertenenciaMax,2));
        m2=pertenenciaMax(:,2:2:size(pertenenciaMax,2));

        %Obtengo los mínimos valores de los máximos. Hago la
traspuesta para
        %guardar los mínimos por columnas.
        minValueMax(:,i)=min(max(m1,m2))';


        %Creo dos matrices, una con los colores 1 de cada fonema y
otra con los
        %colores 2 de cada fonema. Hago esto para luego coger el
máximo de las
        %dos.

        m1=pertenenciaMin(:,1:2:size(pertenenciaMin,2));
        m2=pertenenciaMin(:,2:2:size(pertenenciaMin,2));



        minValueMin(:,i)=min(max(m1,m2))';



        %Inicializo la variable longitudes con tantas filas como
fonemas y una
        %sola columna. Uso el tamaño de minValueMax como podría usar
el de
        %minValueMin.
        longitudes=zeros(size(minValueMax,1),1);
```

```
        %Aumento la longitude de valores > que el umbral en 1 o pongo
a cero
        %las que encuentro por debajo.
        longitudes(minValueMin(:,i) >= minParaContar(1)  &
minValueMax(:,i) >=
minParaContar(2))=longitudes(minValue(:,i)>minParaContar)+1;
        longitudes(minValueMin(:,i) < minParaContar(1)  |
minValueMax(:,i) < minParaContar(2))=0;

        %Comprobamos para cada fonema si cumple que se ha reconocido.
        %Uso el tamaño de minValueMax como podría usar el de
        %minValueMin.
        for k=1:size(minValueMax,1) %el número de fonemas (26)

            %Voy a coger todos los valores consecutivos con min>=50 y
max>=60 para hacer una media
            %en minValueMin y minValueMax, y obener un valor mínimo y
máximo medio del intervalo.
            %El número de valores consecutivos me indica la posible
            %longitud del fonema. Voy a obtener un parámetro de
            %coincidencia de la longitud encontrada con la del fonema.
            %Dependiendo de este valor, se ajustará al valor mínimo o
al
            %máximo del intervalo para mostrarlo como resultado final.
            if longitudes(k) ~= 0
                inicioMedia=i-longitudes(k,1)+1; %la columna en la que
estamos - los valores que llevamos consecutivos.
                finMedia=i;%la columna hasta la que hacemos la media


mediaMinima=mean(minValueMin(k,inicioMedia:finMedia))*100;%media de
los valores entre inicio y fin. Lo paso a valores entre 0 y 100 en
lugar
                %de 0 a 1.

mediaMaxima=mean(minValueMax(k,inicioMedia:finMedia))*100;

                %tipoLongitud() recibe como primer parámetro el tipo
de
                %longitud(vs,s,vl...) que se encuentra en la última
fila, columna 1
                %de cada fonema (MatPhonem), y la longitud con la que
estamos trabajando.

fonemLengMax=longitudTipoEntrenamiento(MatPhonem(size(MatPhonem,1),k*2
-1),longitudes(k),funciones);

fonemLengMin=longitudTipoEntrenamiento(MatPhonem(size(MatPhonem,1)+13,
k*2-1),longitudes(k),funciones);

                %El resultado final de la evaluación del fonema es
resultadoFinal.
                %Se trata de un ajuste del intervalo al valor mínimo o
                %máximo dependiendo de la pertenencia del fonema a esa
                %longitud.
                resultadoFinal=mediaMinima+(fonemLeng*(mediaMaxima-
mediaMinima));
```

```
                %Calculamos el valor final del intervalo ajustándolo
con el valor
                %de la pertenencia de la longitud que tiene cada
fonema.
                resultadoFinal(:,1)=mediaMaxima.*fonemLengMaxima;
                resultadoFinal(:,2)=mediaMinima.*fonemLengMinima;

                 resultadoFinal=mean(resultadoFinal,2);


                %Si el resultado final pasa el umbral establecido.
                if resultadoFinal>=minParaValidar

                    %Guardamos el fonema (número que identifica un
fonema) en la
                    %matriz de resultados.
                    resultados(contRes,1)=k;

                    %Guardamos el porcentaje de parecido
(resultadoFinal) que tiene
                    %el fonema encontrado con el comparado.
                    resultados(contRes,2)=resultadoFinal;

                    %Aumentamos el contador para que se guarde la
siguiente.
                    contRes=contRes+1;

                end
            end
        end

        %Si la matriz de resultados tiene en la posición 1 todavía un
0, quiere
        %decir que no ha encontrado ningún resultado válido.Sin
embargo, si ha
        %encontrado algún resultado, coge el fonema que tenga mayor
proximidad (columna
        %2) y lo guarda en 'y'.
        if resultados(1,1)~=0 %Si ha encontrado algún fonema será como
mínimo el fonema 1.

            % cojo el % con mayor exactitud.
            porcentajeMaximo=max(resultados(:,2));

            %busco en qué posición de la matriz resultados se
encuentra.

fonema=resultados(find(resultados(:,2)==porcentajeMaximo));

            %En la matriz resultados, la posición 'fonema' se
encuentra el
            %verdadero identificador del fonema que queremos guardar.
            y=fonema(1);

        end
        i=i+1;
    end
else
```

```
    if final > 0 %SI SE HA METIDO UN VALOR DESDE EL QUE ACABAR, LA
FUNCIÓN NOS DEVOLVERÁ LA TABLA CON LOS RESULTADOS DE CADA FONEMA


        %Creo la matriz minValue que va a tener un tamaño muy
específico en
        %este caso (26 fonemas, los frames que van desde inicio hasta
final ambos incluidos).
        minValueMin=zeros(size(MatPhonem,2)/2,final);
        minValueMax=zeros(size(MatPhonem,2)/2,final);


        %Variable que indica con que porcentaje de valores de minvalue
se
        %hace la media.
        Average=80;


        %Bucle que recorre todos los frames desde inicio hasta final.
        for i=inicio:final


            %Creamos las 16 matrices de colores para cada frame (8 de
las funciones máximo y otros 8 de funciones mínimo).
            %A partir de las matrices creamos la matriz de
pertenencia.

            mColores=zeros(size(espectrograma,1),8*2);
            for j=1:8

mColores(:,j)=coloresTipoEntrenamiento(j,espectrograma(:,i),funciones)
;

mColores(:,j+8)=coloresTipoEntrenamiento(j+13,espectrograma(:,i),funci
ones);
            end

            pertenenciaMax=MatPhonem(1:size(MatPhonem,1)-1,:);
%Pertenencia tiene todos los fonemas menos la longitud.
            pertenenciaMin=MatPhonem(1:size(MatPhonem,1)-1,:);

            %Creo la matriz de pertenencia para la grabación con los
valores mínimos a partir de las 16 matrices de colores
            %(fila a fila) que hemos creado del "espectrograma Min".
            for k=1:size(pertenenciaMax,1) %Como el espectrograma
tiene 25 bandas este bucle solo se hace 25 veces.

pertenenciaMax(k,pertenenciaMax(k,:)==1)=mColores(k,1);

pertenenciaMin(k,pertenenciaMin(k,:)==1)=mColores(k,9);


pertenenciaMax(k,pertenenciaMax(k,:)==2)=mColores(k,2);

pertenenciaMin(k,pertenenciaMin(k,:)==2)=mColores(k,10);


pertenenciaMax(k,pertenenciaMax(k,:)==3)=mColores(k,3);
```

```
pertenenciaMin(k,pertenenciaMin(k,:)==3)=mColores(k,11);


pertenenciaMax(k,pertenenciaMax(k,:)==4)=mColores(k,4);

pertenenciaMin(k,pertenenciaMin(k,:)==4)=mColores(k,12);


pertenenciaMax(k,pertenenciaMax(k,:)==5)=mColores(k,5);

pertenenciaMin(k,pertenenciaMin(k,:)==5)=mColores(k,13);


pertenenciaMax(k,pertenenciaMax(k,:)==6)=mColores(k,6);

pertenenciaMin(k,pertenenciaMin(k,:)==6)=mColores(k,14);


pertenenciaMax(k,pertenenciaMax(k,:)==7)=mColores(k,7);

pertenenciaMin(k,pertenenciaMin(k,:)==7)=mColores(k,15);


pertenenciaMax(k,pertenenciaMax(k,:)==8)=mColores(k,8);

pertenenciaMin(k,pertenenciaMin(k,:)==8)=mColores(k,16);

            end

            %Creo dos matrices, una con los colores 1 de cada fonema y
otra con los
            %colores 2 de cada fonema. Hago esto para luego coger el
máximo de las
            %dos.

            m1=pertenenciaMax(:,1:2:size(pertenenciaMax,2));
            m2=pertenenciaMax(:,2:2:size(pertenenciaMax,2));

            %Obtengo los mínimos valores de los máximos. Hago la
traspuesta para
            %guardar los mínimos por columnas.
            minValueMax(:,i)=min(max(m1,m2))';


            %Creo dos matrices, una con los colores 1 de cada fonema y
otra con los
            %colores 2 de cada fonema. Hago esto para luego coger el
máximo de las
            %dos.

            m1=pertenenciaMin(:,1:2:size(pertenenciaMin,2));
            m2=pertenenciaMin(:,2:2:size(pertenenciaMin,2));



            minValueMin(:,i)=min(max(m1,m2))';
```

```
        end


        %PARA LONGITUD

        %Guardamos todas las longitudes de cada fonema.

resLongitudesMin=MatPhonem(size(MatPhonem,1),1:2:size(MatPhonem,2))';

resLongitudesMax=MatPhonem(size(MatPhonem,1),1:2:size(MatPhonem,2))';

        %ya tenemos minValue con toda la información. Procedemos a
hacer el
        %AverageN sobre los valores mínimos y los máximos
        resultadosMin=sort(minValueMin(:,inicio:final),2,'descend');

resultadosMin=mean(resultadosMin(:,1:ceil(Average*size(resultadosMin,2
)/100)),2);

        resultadosMax=sort(minValueMax(:,inicio:final),2,'descend');

resultadosMax=mean(resultadosMax(:,1:ceil(Average*size(resultadosMax,2
)/100)),2);


        %Calculamos la pertenencia de la longitud del bloque en cada
una de
        %las longitudes posibles
        for j=1:5
            aLongitudesMax(j)=longitudTipoEntrenamiento(j,final-
inicio+1,funciones);
            aLongitudesMin(j)=longitudTipoEntrenamiento(j+13,final-
inicio+1,funciones);
        end

        resLongitudesMin(resLongitudesMin==1)=aLongitudesMin(1);
        resLongitudesMin(resLongitudesMin==2)=aLongitudesMin(2);
        resLongitudesMin(resLongitudesMin==3)=aLongitudesMin(3);
        resLongitudesMin(resLongitudesMin==4)=aLongitudesMin(4);
        resLongitudesMin(resLongitudesMin==5)=aLongitudesMin(5);

        resLongitudesMax(resLongitudesMax==1)=aLongitudesMax(1);
        resLongitudesMax(resLongitudesMax==2)=aLongitudesMax(2);
        resLongitudesMax(resLongitudesMax==3)=aLongitudesMax(3);
        resLongitudesMax(resLongitudesMax==4)=aLongitudesMax(4);
        resLongitudesMax(resLongitudesMax==5)=aLongitudesMax(5);

        %Calculamos el valor final del intervalo ajustándolo con el
valor
        %de la pertenencia de la longitud que tiene cada fonema.
        resultadoFinal(:,1)=resultadosMax.*resLongitudesMax;
        resultadoFinal(:,2)=resultadosMin.*resLongitudesMin;

        resultadoFinal=mean(resultadoFinal,2);

        %Hago un bucle de forma que trato todos los fonemas.
        for k=1:size(resultadoFinal,1)
            unNumero=find(resultadoFinal==max(resultadoFinal));
```

```
            %En max(MediaFinal) tengo el valor de coincidencia del
fonema.
            %En unNumero tengo la posición donde se encontraba ese
valor, y
            %que corresponde al fonema.
            res(k,1)=max(resultadoFinal);
            res(k,2)=unNumero(1);

            %Pongo el número máximo a -1.
            resultadoFinal(unNumero(1))=-1;
        end

        %Devolvemos la tabla de resultados.
        y=res;

    else
            x='Error, final es un número negativo';
            x
    end
end
```

Funciones del algoritmo genetico

```
function [fonemas funciones]=algoritmoGenetico(matriz,grabacion,func)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
% matriz, grabacion y nombres son 3 variables que se pasan por
parámetros y
% que corresponden a matriz,S y fonemas del fichero variables.mat
% Mas específicamente, son la 3DMatriz de todos los fonemas(matriz),
la
% grabación pasada a mel, y reducida (grab2mel(512) &
speechreduction3(10))
% y el nombre de los fonemas (nombres).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%


%Depende de si introducen una tabla de funciones o no haremos una cosa
u
%otra.
if (func==0)
    %Declaramos la matríz de los valores de las funciones
generalizadas.
    funciones=[0        0       0.05        0.35;
               0        0.05    0.22        0.5;
               0        0.23    0.35        0.63;
               0.05     0.36    0.5         0.8;
               0.23     0.51    0.63        0.93;
               0.36     0.63    0.8         0.99;
```

```
                0.51      0.81      0.93          0.99;
                0.63      0.94      0.99          0.99;

                2         2         3             5;
                2         2         6             10;
                2         6         12            17;
                8         12        20            30;
                12        18        69            99;];
else
    funciones=func;
end

%Declaramos los tiempos en los que empieza cada uno de los distintos
%fonemas de la grabación.
tiemposCorrectos=[1 1; %h#
          13 2; %sh
          18 3; %ix
          23 4; %hv
          27 5; %eh
          35 6; %dcl
          37 7; %jh
          41 8; %ih
          46 6; %dcl
          49 9; %d
          50 10; %ah
          58 11; %kcl
          62 12; %k
          64 13; %s
          71 14; %ux
          80 15; %q
          83 16; %en
          88 17; %gcl
          90 18; %g
          91 19; %r
          95 3; %ix
          100 13; %s
          106 3; %ix
          110 20; %w
          116 21; %ao
          124 2; %sh
          130 22; %epi
          132 20; %w
          135 21; %ao
          141 23; %dx
          142 24; %axr
          146 21; %ao
          154 25; %l
          157 26; %y
          164 8; %ih
          169 24; %axr
          174 1]; %h#

%Depende de si tenemos una matriz 3D o 2D haremos una cosa u otra.
if (size(matriz,3)~=1)
    %Matriz de los fonemas escrita en una sola matriz 2D con la que
trabajaré a
    %partir de ahora.
    fonem=[matriz(:,:,1) matriz(:,:,2) matriz(:,:,3) matriz(:,:,4)
matriz(:,:,5) matriz(:,:,6) matriz(:,:,7) matriz(:,:,8) matriz(:,:,9)
matriz(:,:,10) matriz(:,:,11) matriz(:,:,12) matriz(:,:,13)
matriz(:,:,14) matriz(:,:,15) matriz(:,:,16) matriz(:,:,17)
```

```
matriz(:,:,18) matriz(:,:,19) matriz(:,:,20) matriz(:,:,21)
matriz(:,:,22) matriz(:,:,23) matriz(:,:,24) matriz(:,:,25)
matriz(:,:,26) ];
    mutInicial=400; %Número de mutaciones que diferenciará a un
individuo de otro.
else
    fonem=matriz;
    mutInicial=200; %Número de mutaciones que diferenciará a un
individuo de otro.
    %Como si hemos entrado aquí es porque estamos recuperando una
población
    %guardada, no hacemos mutaciones.
end

%Creo un pool con muchos individuos. Cada individuo está repartido en
dos
%matrices, una con los fonemas y otra con las funciones.La tercera
%dimensión de cada una indica el individuo al que pertenece.

[fonemPool funcPool]=generaPoblacion(fonem,funciones,mutInicial);

evaluacion=zeros(1,size(fonemPool,3));

%obtenemos la evaluación de cada genoma.
for i=1:size(fonemPool,3)
    %Creamos una tabla en la que guardamos la evaluación de cada
individuo.

evaluacion(i)=evaluarPoblacion(fonemPool(:,:,i),grabacion,funcPool(:,:
,i),tiemposCorrectos);
end

seguridad=1;
x=0;
y=[0;0];


%vamos a realizar la mutación, cruce y demás mientras no tengamos un
%resultado óptimo, es decir, mientras no tengamos al menos 18
aciertos.
while max(evaluacion)<=30
    %Después de crear la población, hago los cruces y creo los hijos
duplicando
    %los individuos finales de la población.
    [pobPool fPool]=cruzarPoblacion(fonemPool,funcPool,evaluacion);

    %obtenemos la evaluación de cada genoma.
    for i=1:size(pobPool,3)
        %Creamos una tabla en la que guardamos la evaluación de cada
individuo.

evaluacion(i)=evaluarPoblacion(pobPool(:,:,i),grabacion,fPool(:,:,i),t
iemposCorrectos);
    end

    %Con esto vuelvo a recuperar una población del tamaño de la
inicial pero
    %con los que habían obtenido mejor resultado en la función de
evaluación.
```

```
    [fonemPool funcPool
evaluacion]=nuevaGeneracion(pobPool,fPool,evaluacion);

    matrizRecuperada=fonemPool(:,:,1);
    funcionesRecuperada=funcPool(:,:,1);

    evalMaxima=evaluacion(1);
    evalMinima=evaluacion(size(evaluacion,2));

    %Para escribir la gráfica que indica los máximos y mínimos de cada
    %población.
    x=[x seguridad];
    y(:,size(y,2)+1)=[evalMaxima evalMinima];

    plot(x,y);

    evaluacion
    seguridad=seguridad+1;

    %cada 5 checks guardamos los datos.
    if mod(seguridad,5)==0
        save 'seguridad.mat' matrizRecuperada funcionesRecuperada
evaluacion;
        z='Copia de seguridad hecha';
        z
    end
end

fonemas=fonemPool(:,:,1);
funciones=funcPool(:,:,1);




function [poblacion
funcPob]=generaPoblacion(matriz2d,funciones,numMutaciones)
%Crea una población inicial para el algoritmo genético.

%población es una matriz 3D. En cada tercera dimensión se indica el
%elemento de la población. Cada población es una identificación de los
%fonemas y una tabla de funciones concretas.

%Num elementos de la población
N=100;

%creo una población de N individuos.
poblacion(:,:,1)=matriz2d;
funcPob(:,:,1)=funciones;
for i=2:N

    %Inicializo las poblaciones al azar.
    matriz2d(1:size(poblacion,1)-1,:)=ceil(rand(size(poblacion,1)-
1,size(poblacion,2))*8);

matriz2d(size(poblacion,1),1:2:size(poblacion,2))=ceil(rand(1,size(pob
lacion,2)/2)*5);

    %Guardo el individuo en la posición que le corresponde.
```

```
    poblacion(:,:,i)=matriz2d;
    funcPob(:,:,i)=funciones;
end



function evaluacion=evaluarPoblacion(Matriz, grabacion, funciones,
tiempos)
%Realiza el proceso de reconocimiento calculando un valor según va
%encontrando el fonema que corresponde en una lista.

%Matriz es la matriz 3-Dimensional con cada uno de los fonemas y sus
%máximos, mínimos colores y longitudes.

%grabación es la matriz de la grabación habiendose hecho el paso a mel
y la reducción
%correspondiente de la información. (grab2mel(512) &
%speechreduction3(S,10))

%funciones es la matriz con los valores Start,first,last y end del
calculo
%de los colores y longitudes.

%tiempos es la matriz que indica en que tiempos empieza cada fonema y
el
%número de fonema que encaja en cada parte. Lo usamos para contar el
número
%de aciertos en la búsqueda.

%Vamos a reconocer en cada fragmento de la grabación los diferentes
fonemas
%que tenemos y el porcentaje de acierto que hay en cada uno.

%evaluacion es la variable que dice cuánto de buena es una población.
evaluacion=0;

for i=1:size(tiempos,1)-1

    %resultados es una matriz con los aciertos de cada fonema en este
    %fragmento de la grabación.

resultados=jugarEntrenamiento(Matriz,grabacion,funciones,tiempos(i,1),
tiempos(i+1,1));

    res(:,:,i)=resultados;
    %PosEncontrado es la posición en la que hemos encontrado el
fonema. El
    %find solo nos devuelve una posición, ya que no se va a repetir un
    %fonema.
    PosEncontrado=find(resultados(:,2)==tiempos(i,2));
    if (PosEncontrado==1)
        evaluacion=evaluacion+(1*resultados(PosEncontrado,1));
    elseif (PosEncontrado==2)
        evaluacion=evaluacion+(0.5*resultados(PosEncontrado,1));
    elseif (PosEncontrado==3 || PosEncontrado==4)
        evaluacion=evaluacion+(0.4*resultados(PosEncontrado,1));
    elseif (PosEncontrado>4 && PosEncontrado<8)
        evaluacion=evaluacion+(0.3*resultados(PosEncontrado,1));
```

```
    elseif (PosEncontrado>=8 && PosEncontrado<12)
        evaluacion=evaluacion+(0.15*resultados(PosEncontrado,1));
    end

end



function [fonemPool
funcPool]=cruzarPoblacion(fonemPool,funcPool,evaluacion)
%Esta función genera hijos partiendo de dos padres al azar de toda la
%población. Terminamos de generar los hijos cuando hayamos duplicado
el
%número de individuos totales (size(fonemPool,3)*2)

%hago un bucle que se repita tantas veces como genmas haya (para
%duplicarlos)

tam=size(fonemPool,3);
%Guardo el tamaño inicial de la población porque luego voy a ir
insertando
%valores en fonemPool y por lo tanto el tamaño se vería modificado
(nunca
%saldríamos del bucle.

for i=1:2:tam %para aumentar al doble la población

    ind=1;

    %obtengo índice del genoma primero por torneo
    genom11=extraePos(tam);
    genom12=extraePos(tam);

    vector= [evaluacion(genom11) evaluacion(genom12)];
    pos=find(evaluacion==max(vector));
    if (size(pos,1)>1)
        ind=ceil(rand(1)*size(pos));
    end
    genom1=pos(ind); %Cojo el primero porque puede haber varias
posiciones que tengan ese valor.

    genom2=genom1;
    salida=0;
    while(genom2==genom1 && salida<2)
        %obtengo índice del genoma segundo por torneo
        genom21=extraePos(tam);
        genom22=extraePos(tam);

        vector = [evaluacion(genom21) evaluacion(genom22)];
        pos=find(evaluacion==max(vector));
        if (size(pos,1)>1)
            ind=ceil(rand(1)*size(pos));
        end
        genom2=pos(ind); %Cojo el primero porque puede haber varias
posiciones que tengan ese valor.

        salida=salida+1;
    end
```

```
    %por cada cruce obtengo dos hijos
    [newFonem1 newFunc1 newFonem2
newFunc2]=cruzar(fonemPool(:,:,genom1),funcPool(:,:,genom1),fonemPool(
:,:,genom2),funcPool(:,:,genom2));

    %Muto los nuevos 2 fonemas hijos 2 veces.
    [newFonem1 newFunc1]=mutar(newFonem1,newFunc1);
    [newFonem1 newFunc1]=mutar(newFonem1,newFunc1);


    [newFonem2 newFunc2]=mutar(newFonem2,newFunc2);
    [newFonem2 newFunc2]=mutar(newFonem2,newFunc2);


    %guardo el nuevo genoma1.
    fonemPool(:,:,tam+i)=newFonem1;
    funcPool(:,:,tam+i)=newFunc1;

    %guardo el nuevo genoma2.
    fonemPool(:,:,tam+i+1)=newFonem2;
    funcPool(:,:,tam+i+1)=newFunc2;
end



function [poblacion1 funcion1 poblacion2
funcion2]=cruzar(pob1,func1,pob2,func2)
%Esta función escoge al azar que partes pertenecerán a un hijo y
cuales a
%otro hijo.

%trato cada fila de las funciones
for i=1:size(func1,1)
    %elijo el padre1 para la población 1
    if rand()>=0.5
        funcion1(i,:)=func1(i,:);
        funcion2(i,:)=func2(i,:);
    %elijo el padre2 para la población 2
    else
        funcion1(i,:)=func2(i,:);
        funcion2(i,:)=func1(i,:);
    end
end

%trato cada fonema (colores)
for i=1:2:size(pob1,2) %desde el fonema 1 hasta el 52 de 2 en 2 ( = 26
fonemas)
    %elijo el padre1 para la población 1
    if rand()>=0.5
        %Esta vez copio el fonema entero y este consta de dos columnas
que
        %lo identifican (tanto la i como la i+1).
        poblacion1(:,i)=pob1(:,i);
        poblacion1(:,i+1)=pob1(:,i+1);

        poblacion2(:,i)=pob2(:,i);
        poblacion2(:,i+1)=pob2(:,i+1);
    %elijo el padre2 para la población 2
```

```
    else
        poblacion1(:,i)=pob2(:,i);
        poblacion1(:,i+1)=pob2(:,i+1);

        poblacion2(:,i)=pob1(:,i);
        poblacion2(:,i+1)=pob1(:,i+1);
    end
end
```

(Para caso de artículo y datos)

```
function [matriz funciones]=mutar(matriz,funciones)
%Esta función coge la matriz con todos los fonemas y la tabla de
funciones
%y les aplica una sola mutación a algún valor de cualquiera de las
dos.

tam=(size(matriz,2)/2)+size(funciones,1); %el tamaño máximo será la
suma de los fonemas
%más las filas de funciones (colores y longitud). En total 26 + 13 =
39
linea=extraePos(tam); %porque también puede salir 0.

%Ya obtenida la línea que queremos mutar tenemos que comprobar a que
%correpsonde.

if linea < 9 %si es de la 1 a la 13 se trata de una mutación en las
funciones
    %así que genero la posición de la función que muta.
(Start,First,Last o
    %End) Como los valores máximos de las longitudes van de 0 a 99 y
no de
    %0.0 a 0.99 las trato diferente.
    pos=extraePos(4); %hay 4 posibles valores a mutar (tmb sale el 0).

    %para ver si sumo o resto.
    if rand() >= 0.5 %sumo
        if funciones(linea,pos)~=0.99
            funciones(linea,pos)=funciones(linea,pos)+0.01;
        end
    else %resto
        if funciones(linea,pos)~=0
            funciones(linea,pos)=funciones(linea,pos)-0.01;
        end
    end

    %debemos reordenar los valores de la función por si se han
cambiado los
    %máximos.
    funciones(linea,:)=sort(funciones(linea,:));
elseif linea < 14
    pos=extraePos(4); %hay 4 posibles valores a mutar (tmb sale el 0).

    %para ver si sumo o resto.
    if rand() >= 0.5 %sumo
        if funciones(linea,pos)~=99
```

```
            funciones(linea,pos)=funciones(linea,pos)+1;
        end
    else %resto
        if funciones(linea,pos)~=0
            funciones(linea,pos)=funciones(linea,pos)-1;
        end
    end

    %debemos reordenar los valores de la función por si se han
cambiado los
    %máximos.
    funciones(linea,:)=sort(funciones(linea,:));
elseif linea < 40 % se trata de un fonema
    % linea indica el fonema.

    % Calculo la posición del fonema entre 1 y 27 (la 27 es la
longitud)
    pos=extraePos(27);

    if pos==27 %si tengo que mutar la longitud no hace falta mucho mas
        if rand()>=0.5 %sumo 1 a la longitud si es distinto de la
máxima longitud(vl)
            if matriz(pos,(linea-13)*2-1)~=5
                %es linea*2-1 porque asi se accede al fonema en una
matriz-2D
                %es (linea-13) porque el primer fonema se encontrará
en el
                %valor 14 de línea, ya que del 1 al 13 son para la
tabla
                %funciones
                matriz(pos,(linea-13)*2-1)=matriz(pos,(linea-13)*2-
1)+1;
            end
        else %resto 1 a la longitud si es distinta de 1 (vs)
            if matriz(pos,(linea-13)*2-1)~=1
                %es linea*2-1 porque asi se accede al fonema en una
matriz-2D
                matriz(pos,(linea-13)*2-1)=matriz(pos,(linea-13)*2-1)-
1;
            end
        end
    %para decidir si muto el primer color o el segundo hago otro rand.
    elseif rand()>=0.5 %columna1
        if rand()>=0.5 %sumo 1 al color correspondiente (Ej:paso de
negro a azul..)
            if matriz(pos,(linea-13)*2-1)~=8 %8 es el máximo color
                matriz(pos,(linea-13)*2-1)=matriz(pos,(linea-13)*2-
1)+1;
            end
        else
            if matriz(pos,(linea-13)*2-1)~=1 %8 es el máximo color
                matriz(pos,(linea-13)*2-1)=matriz(pos,(linea-13)*2-1)-
1;
            end
        end
    else %columna2 (solo cambia el acceso a la matriz, en lugar de
(linea-13)*2-1 es sin "-1"
        if rand()>=0.5 %sumo 1 al color correspondiente (Ej:paso de
negro a azul..)
            if matriz(pos,(linea-13)*2)~=8 %8 es el máximo color
                matriz(pos,(linea-13)*2)=matriz(pos,(linea-13)*2)+1;
```

```
            end
        else
            if matriz(pos,(linea-13)*2)~=1 %8 es el máximo color
                matriz(pos,(linea-13)*2)=matriz(pos,(linea-13)*2)-1;
            end
        end

    end
else
    x='Error, se intenta mutar algo no alcanzable';
    x
    linea
end
```

(Para caso de intervalos en funciones)

```
function [matriz funciones]=mutar(matriz,funciones)
%Esta función coge la matriz con todos los fonemas y la tabla de
funciones
%y les aplica una sola mutación a algún valor de cualquiera de las
dos.

tam=(size(matriz,2)/2)+size(funciones,1)/2; %el tamaño máximo será la
suma de los fonemas
%más las filas de funciones (colores y longitud). En total 26 + 13 =
39
linea=extraePos(tam);

%Ya obtenida la línea que queremos mutar tenemos que comprobar a que
%correpsonde.

if linea < 9 %si es de la 1 a la 13 se trata de una mutación en las
funciones
    %así que genero la posición de la función que muta.
(Start,First,Last o
    %End de la primera dimensión, y Start,First,Last o End de la
segunda dimensión)
    %Como los valores máximos de las longitudes van de 0 a 99 y no de
    %0.0 a 0.99 las trato diferente.
    pos=extraePos(8); %hay 8 posibles valores a mutar.

    %Para determinar la dimensión de las funciones que voy a mutar.
    if (pos>4) %Modifico las funciones de mínimo

         %para ver si sumo o resto.
        if rand() >= 0.5 %sumo
            if funciones(linea+13,mod(pos,5)+1)~=0.99

funciones(linea+13,mod(pos,5)+1)=funciones(linea+13,mod(pos,5)+1)+0.01
;
            end
        else %resto
            if funciones(linea+13,mod(pos,5)+1)~=0

funciones(linea+13,mod(pos,5)+1)=funciones(linea+13,mod(pos,5)+1)-
0.01;
```

```
            end
        end

        %debemos reordenar los valores de la función por si se han
cambiado los
        %máximos.
        funciones(linea+13,:)=sort(funciones(linea+13,:));
    else
        %para ver si sumo o resto.
        if rand() >= 0.5 %sumo
            if funciones(linea,pos)~=0.99
                funciones(linea,pos)=funciones(linea,pos)+0.01;
            end
        else %resto
            if funciones(linea,pos)~=0
                funciones(linea,pos)=funciones(linea,pos)-0.01;
            end
        end

        %debemos reordenar los valores de la función por si se han
cambiado los
        %máximos.
        funciones(linea,:)=sort(funciones(linea,:));
    end

    %Para que no tenga la función máximo valores menores que la
función
    %mínimo tengo que reordenarlas.

    auxiliar=[funciones(linea,:); funciones(linea+13,:)];

    auxiliar=sort(auxiliar,'descend'); %Al hacer descend se va a
quedar en auxiliar(1,:) los valores máximos

    funciones(linea,:)=auxiliar(1,:); %1 son los valores de las
funciones maximos
    funciones(linea+13,:)=auxiliar(2,:); %2 son los valores de las
funciones mínimos

elseif linea < 14
    pos=extraePos(8); %hay 8 posibles valores a mutar.

    %Para determinar la dimensión de las funciones que voy a mutar.
    if (pos>4) %Modifico las funciones de mínimo

         %para ver si sumo o resto.
        if rand() >= 0.5 %sumo
            if funciones(linea+13,mod(pos,5)+1)~=99

funciones(linea+13,mod(pos,5)+1)=funciones(linea+13,mod(pos,5)+1)+1;
            end
        else %resto
            if funciones(linea+13,mod(pos,5)+1)~=0

funciones(linea+13,mod(pos,5)+1)=funciones(linea+13,mod(pos,5)+1)-1;
            end
        end
```

```
        %debemos reordenar los valores de la función por si se han
cambiado los
        %máximos.
        funciones(linea+13,:)=sort(funciones(linea+13,:));
    else
        %para ver si sumo o resto.
        if rand() >= 0.5 %sumo
            if funciones(linea,pos)~=99
                funciones(linea,pos)=funciones(linea,pos)+1;
            end
        else %resto
            if funciones(linea,pos)~=0
                funciones(linea,pos)=funciones(linea,pos)-1;
            end
        end

        %debemos reordenar los valores de la función por si se han
cambiado los
        %máximos.
        funciones(linea,:)=sort(funciones(linea,:));
    end

    %Para que no tenga la función máximo valores menores que la
función
    %mínimo tengo que reordenarlas.

    auxiliar=[funciones(linea,:); funciones(linea+13,:)];

    auxiliar=sort(auxiliar,'descend'); %Al hacer descend se va a
quedar en auxiliar(1,:) los valores máximos

    funciones(linea,:)=auxiliar(1,:); %1 son los valores de las
funciones maximos
    funciones(linea+13,:)=auxiliar(2,:); %2 son los valores de las
funciones mínimos

elseif linea < 40 % se trata de un fonema
    % linea indica el fonema.

    % Calculo la posición del fonema entre 1 y 27 (la 27 es la
longitud)
    pos=extraePos(27);

    if pos==27 %si tengo que mutar la longitud no hace falta mucho mas
        if rand()>=0.5 %sumo 1 a la longitud si es distinto de la
máxima longitud(vl)
            if matriz(pos,(linea-13)*2-1)~=5
                %es linea*2-1 porque asi se accede al fonema en una
matriz-2D
                %es (linea-13) porque el primer fonema se encontrará
en el
                %valor 14 de línea, ya que del 1 al 13 son para la
tabla
                %funciones
                matriz(pos,(linea-13)*2-1)=matriz(pos,(linea-13)*2-
1)+1;
            end
        else %resto 1 a la longitud si es distinta de 1 (vs)
            if matriz(pos,(linea-13)*2-1)~=1
```

```
                %es linea*2-1 porque asi se accede al fonema en una
matriz-2D
                matriz(pos,(linea-13)*2-1)=matriz(pos,(linea-13)*2-1)-
1;
            end
        end
    %para decidir si muto el primer color o el segundo hago otro rand.
    elseif rand()>=0.5 %columna1
        if rand()>=0.5 %sumo 1 al color correspondiente (Ej:paso de
negro a azul..)
            if matriz(pos,(linea-13)*2-1)~=8 %8 es el máximo color
                matriz(pos,(linea-13)*2-1)=matriz(pos,(linea-13)*2-
1)+1;
            end
        else
            if matriz(pos,(linea-13)*2-1)~=1 %8 es el máximo color
                matriz(pos,(linea-13)*2-1)=matriz(pos,(linea-13)*2-1)-
1;
            end
        end
    else %columna2 (solo cambia el acceso a la matriz, en lugar de
(linea-13)*2-1 es sin "-1"
        if rand()>=0.5 %sumo 1 al color correspondiente (Ej:paso de
negro a azul..)
            if matriz(pos,(linea-13)*2)~=8 %8 es el máximo color
                matriz(pos,(linea-13)*2)=matriz(pos,(linea-13)*2)+1;
            end
        else
            if matriz(pos,(linea-13)*2)~=1 %8 es el máximo color
                matriz(pos,(linea-13)*2)=matriz(pos,(linea-13)*2)-1;
            end
        end

    end
else
    x='Error, se intenta mutar algo no alcanzable';
    x
    linea
end
```

```
function [nuevaPob nuevaF
ev]=nuevaGeneracion(pobPool,fPool,evaluacion)
%Función que ordena las poblaciones según su evaluación para quedarse
con
%los primeros mejores individuos (según el valor de su evaluación).

%Ordeno el vector de evaluación
result=sort(evaluacion,2,'descend');

for i=1:size(pobPool,3)/2 %guardo la mitad de los valores que tenía.

    %busco en evaluación el índice que corresponde con el mayor valor.
    indice=find(evaluacion==result(i));
    indice=indice(1); %me quedo con el primero por si se ha encontrado
mas de uno.

    evaluacion(indice)=-1; %borro el valor examinado.
```

```
        %Guardo el genoma correspondiente a ese índice.
        nuevaPob(:,:,i)=pobPool(:,:,indice);
        nuevaF(:,:,i)=fPool(:,:,indice);
        ev(i)=result(i);
    end
```